NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

AN INVESTIGATION OF LOW MARANGONI NUMBER FLUID FLOW IN A COLD CORNER

by

Michael R. Huber

June 1993

Thesis Advisor: D. Canright

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 1993	Master's Thesis

4. TITLE AND SUBTITLE
AN INVESTIGATION OF LOW MARANGONI NUMBER FLUID FLOW IN A COLD CORNER

5. FUNDING NUMBERS

6. AUTHOR(S)
HUBER, Michael R.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

A large pool of liquid with a horizontal free surface in bounded on one side by a vertical solid wall. The wall is maintained at a cold temperature to a depth of unity, with a warmer temperature below that point. The fluid surface is assumed adiabatic, and average surface tension forces keep the surface flat. Surface tension is assumed to be a decreasing function of temperature, so that the surface thermal gradient associated with the temperature variations drives flow toward the corner. This problem is examined numerically for different Marangoni numbers ranging from 1 to 300 using a Green's function approximation method for viscous case (in the limit as the Reynolds number approaches zero).

14. SUBJECT TERMS	15. NUMBER OF PAGES
Marangoni Number, Convection, Diffusion, Alternating-Direction Implicit (ADI) Method.	91
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

An Investigation of Low Marangoni Number Fluid Flow in a Cold Corner

by

Michael R. Huber
Captain, United States Army
B.S., Loyola College, 1982
M.S.E., The Johns Hopkins University, 1984

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN APPLIED MATHEMATICS

from the

NAVAL POSTGRADUATE SCHOOL
June 1993

ABSTRACT

A large pool of liquid with a horizontal free surface is bounded on one side by a vertical solid wall. The wall is maintained at a cold temperature to a depth of unity, with a warmer temperature below that point. The fluid surface is assumed adiabatic, and average surface tension forces keep the surface flat. Surface tension is assumed to be a decreasing function of temperature, so that the surface thermal gradient associated with the temperature variations drives flow toward the corner. This problem is examined numerically for different Marangoni numbers ranging from 1 to 300 using a Green's function approximation method for the viscous case (in the limit as the Reynolds number approaches zero).

DISCLAIMER

The computer programs in the appendices are supplied on an "as is" basis, with no warrantees of any kind. The author bears no responsibility for any consequences of using the programs.

CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

This work was supported by the Office of Naval Research, Materials Division (contract N0001492WR24009). I wish to thank my advisor, Dr. David Canright, for his expert assistance and dynamic guidance which made this project a tremendous learning and enjoyable experience. I also wish to thank Dr. Van E. Henson, for his untiring and sincere assistance with my many numerical analysis questions, and Dr. Clyde Scandrett, for his helpful insights as second reader. Finally, I wish to thank my wife, Teresa, and my children, Nikolas, Kirstin, and Stephanie, for their enthusiastic and unending support throughout the past two years. I dedicate this work to them.

I. INTRODUCTION

This project is meant to increase the understanding of a particular boundary value problem with real-world applications. The phenomenon of thermocapillary convection has an important role in fluid flow, especially in welding processes, where convection in the molten metal can affect the microstructure of the material. A major tool in this project is using the Green's function to solve the partial differential equation (PDE). This work combines the use of a Green's function (for an infinite domain) with a numerical scheme to solve an elliptic PDE. The Marangoni number M, which gives the relative importance of thermal convection to thermal diffusion, will be incremented in the Conservation of Energy equation over a broad range to study how flow affects the temperature field in the cold corner. When M is small, the temperature field is conductive and the flow is dominated by viscous forces; large M corresponds to a vigorous flow with thermal convection becoming dominant. *Mathematica* and *MATLAB* were chosen as the software tools, due to their versatile mathematics capabilities.

A. MOTIVATION

The role of thermocapillary convection is important in the fluid flow and heat transfer of a number of materials processing techniques, notably welding, crystal growth, and other processes involving concentrated energy beams [Ref. 1]. Typically,

these processes involve a pool of molten material with severe temperature gradients along the surface, and it is the surface tension gradient associated with these temperature variations that drives the flow.

Thermocapillary convection refers to the heat transfer/fluid mechanics phenomenon driven by surface tension gradients when associated with surface temperatures non-uniform in nature. Major consequences of this type of convection are: (a) a change of temperature distribution and hence the weld's melt pool shape, (b) solidification and cooling rates of the melt pool, and, in turn, (c) a variation in the final product microstructure. Convection increases the overall transport and growth rate, which is desirable. However, it also seems to affect the morphology of the solid adversely [Ref. 2], since the characteristics of the solid are determined by what occurs near the fluid-solid interface. Convection can change the solid's composition across this interface, through heat release, density change, and other processes, causing nonuniformities which in turn change the solid's crystal structure and shape.

B. PHYSICS OF THE PROBLEM

Convection in the molten pool is usually vigorous, with results outlined above. The predominant forces driving convection are thermocapillary forces, which include changes in surface tension with temperature along the surface of the molten pool.

1. Surface Tension

An understanding of surface tension and its effects is therefore key in the formation of and flow in the weld pool [Ref. 3]. Within the body of a liquid or

solid, the net force on any given atom or molecule is relatively small: it is surrounded by a group of other atoms which exert forces in all directions so that there is little or no resultant. At the surface, however, there is a resultant attraction inwards because the molecular density is much higher in the liquid than in the surrounding air (or vapor). Because of this inward force, the surface of a liquid tends to contract to the smallest possible size, so that drops and bubbles will, in the absence of external constraints, become spherical in form. Work must be done in order to bring a molecule from the bulk of the liquid to the surface against the inwardly-acting forces, and the work required to produce one unit of new area in this way is called the *free surface energy*, or *free energy*. As a result of this tendency of the surface to contract, it may be considered as in a state of tension. The *surface tension* is the force acting along a unit length of a liquid surface.

There are two basic modes of flow generated by surface tension gradients. When there is a gradient of surface tension along (parallel to) an interface, a shear stress is generated and this may generate flow or affect existing flow. Such flows were first investigated by Marangoni (1871) and the effect is sometimes referred to as "Marangoni flow" [Ref. 3]. If the gradient is perpendicular to the interface, a "Marangoni instability" can occur, leading to cellular flows [Ref. 2].

The surface tension of a liquid generally decreases with increasing temperature. If part of the free surface should become locally hotter than the rest, as a result of some small disturbance, fluid is drawn away from the region by the action of

surface tension. In the welding process, the heat absorbed by the substrate raises the temperature and develops a molten pool. The surface temperature decreases radially outward from the center of this pool (where the energy beam is strongest) [Ref. 4]. The surface tension thus increases radially outward from the center, and as the flow develops, the energy transfer mechanism becomes convective and the fluid flow is *driven* by the surface tension gradient. At the liquid-air interface, the pressure force must be balanced by the surface tension, and is not necessarily zero as it must be at a free surface without surface tension [Ref. 5]. A finite pressure difference can exist across the free surface interface and be balanced by the surface tension. When the fluid at the surface is pulled in the direction of increasing surface tension, the ensuing motion creates a normal gradient of the tangential velocity [Ref. 1]. The motion is also accompanied by a shear stress generated by the Marangoni flow, which balances the pressure force due to this surface tension gradient. This shear stress provides a balance, since the fluid motion is often caused by varying temperature distributions in the fluid, which are associated with surface tension non-uniformities. In turn, the shear stress acts on the fluid in the interior of the molten pool, setting up a bulk convective motion.

2. Thermocapillary Flow

In a heat transfer system, the measure of intensity of convection relative to conduction is the *Peclet* number, and the Peclet number based on a thermocapillary velocity is the *Marangoni* number, designated in this paper as M, which is the basic

dimensionless parameter of thermocapillary convection. M measures the strength of temperature (thermal energy) convection relative to diffusion. Thus, large values of M will often lead to the formation of thermal boundary layers. Chen associates thermocapillary flow encountered in materials processing with high Marangoni number [Ref. 1]. For many practical materials processing operations, the Marangoni number is of the order of 10^4 or above.

3. Thermal Convection

Consider a two-dimensional rectangular weld pool, with one hot wall and one cold wall, each perpendicular to the weld surface (as considered by [Ref. 6]). Numerical evaluation of the local heat flux shows a concentration of the thermal gradient near the cold corner [Ref. 1, 6]. As M increases, the temperature distribution gradually changes from a linear distribution (characteristic of conduction solutions) to one characteristic of high Peclet number convection. The cold corner velocity distribution is apparently due to the fact that the intense convection associated with the surface motion toward the cold wall has brought the warm fluid forward, significantly compressing the region of temperature variation; this in turn intensifies the local thermocapillary forces driving the flow. In between the hot and cold walls is a region of constant surface temperature. The flow then can be viewed as a half-jet [Ref. 1], maintained by inertia, issuing from the hot corner.

In typical applications, the cold wall represents the solid-liquid phase boundary. Hence, any concentrated heat flux would affect the shape of this melt-

ing/solidification interface, i.e., changing the shape of the molten weld pool. The non-uniformity of the heat flux is not fully understood, which led Chen to state, "It would seem then that the structure of the cold corner flow is one of the most critical issues to be studied in the future" [Ref. 1]. This research concentrates on effects of temperature and flow in the cold corner region.

a. Historical Note

Historically, Ludwig Prandtl credits Osborne Reynolds to have been the first who clearly recognized the part played in heat transfer by the velocity of flow [Ref. 7]. In short papers published in 1874 [Ref. 8] and again in 1900 [Ref. 9], Reynolds points out that the theories on heat transfer up to that time, in which conductivity alone had been taken into account, must be in error and therefore revised. Reynolds asserts that the main phenomenon in heat transfer is that particles move from the interior of the fluid up to the boundary and bring their heat with them. The resistance to flow arises in the same way as a result of the particles bringing their velocities from the interior of the fluid up to the boundary or surface, where they give rise to frictional forces. Unaware of Reynolds' work, Prandtl rediscovered the same train of thought, but in a more accurate mathematical form, in 1910 [Ref. 10]. Both Reynolds' and Prandtl's arguments are now referred to as the *momentum theory of heat transfer* [Ref. 7].

C. PREVIOUS RESEARCH

Comparing recent studies, Chan, Mazumder, and Chen [Ref. 4] present a three-dimensional model for thermocapillary convection during surface melting due to a moving laser heat source. They use a perturbation solution to model the three-dimensional flow with two sets of two-dimensional equations, instead of a more complicated three-dimensional set of equations. The detailed three-dimensional velocity field gives a quantitative explanation of the mechanism of the mixing process within the molten pool. In their calculations, the surface of the melt pool is assumed to be flat to simplify the surface boundary conditions. Using the energy equation in cylindrical coordinates within the molten pool, they derive finite-difference equations, central-difference for the diffusion terms, and upwind-difference for the convective terms (governing the velocity and temperature) and they solve the energy equation by the alternating-direction iteration method. Then, the velocities within the molten pool are time-step iterated for a prescribed number of iterations. Using this updated velocity field, the energy equations are iterated next. Thus, the temperature field for the global iteration is obtained. The initial guesses are the steady-state conduction temperature and zero velocity for the perturbation solution. They find that the presence of the thermocapillary convection changes the physics of the process from conduction to convection dominated.

Cowley and Davis [Ref. 11] analyze the thermocapillary flow near the hot wall for vigorous flow, formulating a canonical $M \rightarrow \infty$ convection problem near the hot

wall for large M values. This involves a distinguished limit where both $M \to \infty$ and the Prandtl number $P \to \infty$ in a relative way. They find that the fluid flows up the hot wall and away along the free surface.

Zebib, Homsy, and Meiburg [Ref. 6] compute steady thermocapillary flows in a two-dimensional square pool (i.e., with an aspect ratio of unity) by a finite difference procedure and find that the vorticity is discontinuous at the hot and cold corners and will assume different values as a corner is approached on different paths. Boundary layer formation is observed for large M values. Their numerical evaluation of the local heat flux along the cold boundaries shows that indeed, most of the heat transfer occurs near the cold stagnation point. This contrasts with the heat flux distributions along the hot wall, which are evenly distributed. There is also a singularity at the cold corner which is rather unique for the thermocapillary flow. In addition, a more serious problem of discretization error, in this case shared by both the finite difference and finite element methods, is the false diffusion caused by the popular upwind-differencing scheme.

II. PROBLEM STATEMENT

Figure 1 shows the generic weld picture of an incompressible Newtonian liquid, with a heat source penetrating and creating a molten pool within the metal solid. As an idealized problem, consider a magnified view of the three-phase junction ("zooming in" to where the air, solid, and liquid meet), which is bounded on the left and top. To the left is a vertical solid wall, and above the liquid is a horizontal free surface, thus creating a semi-infinite domain (quarter-plane, see Figure 2). The wall is kept at a constant cold temperature, designated T_c, to a depth $d = 1$. (This unit depth is equivalent after rescaling to any other depth). Below this point, the rest of the wall is at the hotter ambient temperature of the surrounding fluid, designated T_h. The surface tension is assumed to be a decreasing function of the temperature, and to be strong enough to keep the surface flat. The surface tension gradients drive the fluid flow, which is assumed steady and two-dimensional, toward the upper left cold corner. The flat free surface is thermally insulated.

Then the equations governing the thermocapillary convection in the cold corner are conservation of mass, momentum, and energy:

$$\nabla \cdot \mathbf{u} = 0 \tag{II.1}$$

$$\rho \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \mu \nabla^2 \mathbf{u} \tag{II.2}$$

$$\rho c_p \mathbf{u} \cdot \nabla T = k \nabla^2 T \tag{II.3}$$

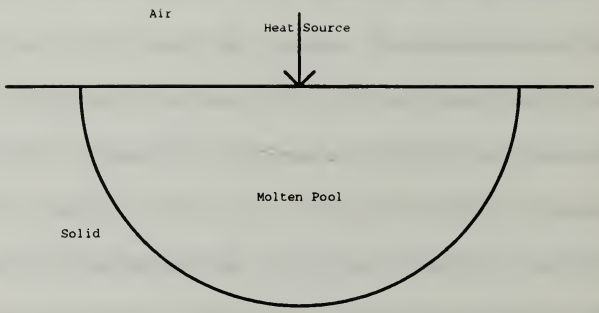

Figure 1. Generic Problem Formulation: A heat source creates a large molten pool of an incompressible Newtonian liquid bounded above by air and beneath by the solid metal.

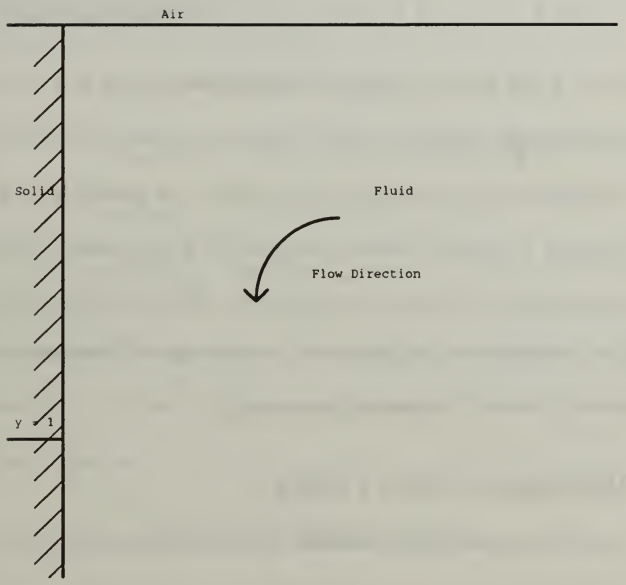

Figure 2. Problem Formulation with Boundary Conditions: A semi-infinite quarter-plane is defined above by a horizontal free surface ($T_y = 0$) and to the left by a vertical solid wall. The wall is cooled ($T = T_c$) to a depth $y = 1$. Below this point, $T = T_h$, the same warmer temperature as that of the pool fluid far away from the wall and surface.

with the following boundary conditions:

$$\text{as } x, y \to \infty: \qquad T \to T_h, \quad u, v \to 0 \tag{II.4}$$

$$\text{at } y = 0: \qquad T_y = 0, \quad v = 0, \quad \mu u_y = \gamma T_x \tag{II.5}$$

$$\text{at } x = 0: \qquad T = T_c, \ (y < d), \ T = T_h \ (y > d), \quad u = v = 0 \tag{II.6}$$

Here $\mathbf{u}$ is the velocity vector with components $u(x,y)$ and $v(x,y)$ in the x direction (horizontally rightward) and y direction (vertically downward), p is pressure, T is temperature, ρ is density, μ is viscosity, c_p is specific heat, k is thermal conductivity, and γ (assumed constant and positive) is the negative of the derivative of the surface tension with respect to temperature. The boundary conditions specify that the wall is piecewise isothermal with no fluid slip, and the flat free surface is thermally insulated, with thermocapillary forcing.

A. DIMENSIONLESS FORM

To nondimensionalize the equations, use the following scale factors:

$$\mathbf{u}' = \mathbf{u}/u_s, \quad T' = \frac{T - T_h}{T_c - T_h}, \quad \mathbf{x}' = \mathbf{x}/d, \quad t' = \frac{t u_s}{d} \tag{II.7}$$

where $u_s = \gamma T/\mu$ is a representative velocity, the temperature difference is $T_c - T_h$, and the representative length is d. Dropping primes, the energy equation simplifies to

$$\frac{u_s d}{\kappa}\, \mathbf{u} \cdot \nabla T = \nabla^2 T \tag{II.8}$$

where $\mathbf{u}$ and T are now dimensionless quantities. Here, $(u_s d) \,/\, \kappa$ is defined to be the dimensionless parameter M, the Marangoni number, so the energy equation becomes:

$$\nabla^2 T = M\, \mathbf{u} \cdot \nabla T \tag{II.9}$$

with boundary conditions:

$$\begin{aligned}
&\text{at } x = 0: && T = -1, \text{ when } y < 1, \quad u = v = 0 \\
&\text{at } x = 0: && T = 0, \text{ when } y > 1, \quad u = v = 0 \\
&\text{at } y = 0: && T_y = 0, \quad u_y = T_x, \quad v = 0 \\
&\text{as } x, y \to \infty: && T \to 0, \quad u, v \to 0
\end{aligned} \tag{II.10}$$

This is a convection-diffusion equation. We seek to find an expression for the steady temperature $T(x, y)$. In the initial case, with Marangoni number M set to zero, we have Laplace's equation

$$\nabla^2 T = T_{xx}(x, y) + T_{yy}(x, y) = 0 \tag{II.11}$$

Now, to nondimensionalize the Momentum Equation, in addition to the previous scaling factors, let:

$$p' = \frac{d(p - p_0)}{\mu u_s}, \tag{II.12}$$

where p_0 is some representative value of the modified pressure in the fluid. Then the steady-state Momentum Equation becomes (in tensor component notation):

$$u'_j \frac{\partial u'_i}{\partial x'_j} = \frac{1}{R}\left[-\frac{\partial p'}{\partial x'_i} + \frac{\partial^2 u'_i}{\partial x'_j \partial x'_j}\right] \tag{II.13}$$

in which the Reynolds number $R = (\rho d u_s)/\mu = (d u_s)/\nu$, where $\nu = \mu/\rho$ is the kinematic viscosity. Dropping the primes gives (in vector notation):

$$R\,\mathbf{u}\cdot\nabla\mathbf{u} = -\nabla p + \nabla^2\mathbf{u} \tag{II.14}$$

where $\mathbf{u}$ and p are now the nondimensional velocity and pressure. The mass equation II.1 retains the same form in nondimensional variables.

B. VISCOUS LIMIT: $R \to 0$

The presence of the non-linear term $\mathbf{u}\cdot\nabla\mathbf{u}$ makes solution of the equation of momentum very difficult for any but the simplest flow fields [Ref. 12]. When inertia is negligible everywhere in the flow field (i.e., $R \to 0$), the Momentum Equation becomes

$$\nabla p = \nabla^2\mathbf{u} \tag{II.15}$$

In this limit, the flow equations are linear, allowing the use of a variety of standard techniques, in particular the Green's function method. Note that $R \to 0$ implies a material with a large Prandtl number, so the method doesn't apply directly to metals. Then, with no inertia, the flow everywhere depends only on the instantaneous thermal gradient along the surface (even if the flow is unsteady). Thus the solution satisfies the boundary conditions and equations of motion with inertia forces neglected. The components of acceleration of the fluid at any point, evaluated according to this solution, are proportional to u_s^2/d [Ref. 12]. The viscous forces, also evaluated according

to this solution, are of order $(\mu u_s)/d^2$, so that the assumption of negligible inertia forces is self-consistent if $(\rho u_s d)/\mu \ll 1$ $(R \ll 1)$.

C. STREAM FUNCTION AND VORTICITY

Since the boundary conditions involve $\mathbf{u}$ alone, the pressure can be eliminated by taking the curl of (II.15), and the problem is to find the solution to

$$\nabla \cdot \mathbf{u} = 0 \tag{II.16}$$

$$\nabla^2(\nabla \times \mathbf{u}) = \nabla^2 \omega = 0 \tag{II.17}$$

where $\omega = \nabla \times \mathbf{u}$ is the vorticity. In the present case of two-dimensional motion with negligible inertia forces, it is convenient to introduce a stream function Ψ so that the conservation of mass equation is satisfied identically and the single non-zero component of vorticity becomes $\omega = -\nabla^2 \Psi$. Then, $\nabla^2(\nabla \times \mathbf{u}) = 0$ becomes

$$\nabla^2(\nabla^2 \Psi) = 0 \tag{II.18}$$

which is the biharmonic equation. Here Ψ is the stream function, such that the velocity components (u, v) are given by $u = \Psi_y$, $v = -\Psi_x$. A significant advantage in using a single scalar function Ψ formulation for the flow is that a single PDE is obtained, instead of coupled PDEs for the pressure p and velocity components u and v. The trade-off for this advantage is that now the partial differential equation is of a higher order.

III. GREEN'S FUNCTION METHOD

As stated earlier, this project concentrates on the effect of temperature and velocity in the cold corner region at various Marangoni numbers. The problem is studied using a Green's function approach to represent the flow. An understanding of the theory behind the Green's function is relevant here, and an example in one dimension is given. The Green's function is a response to a boundary value problem when a forcing function represents a concentrated unit source and the boundary conditions are homogeneous. This is followed by the specific application of the use of the Green's function to the problem.

A. EXAMPLE IN ONE DIMENSION

As an example, consider first the problem consisting of the ordinary differential equation

$$\mathcal{L}\,\Phi + f(x) = 0, \tag{III.1}$$

where $\mathcal{L}$ is the differential operator

$$\mathcal{L} = \frac{d}{dx}\left(p\frac{d}{dx}\right) + q = p\frac{d^2}{dx^2} + \frac{dp}{dx}\frac{d}{dx} + q, \tag{III.2}$$

together with *homogeneous* boundary conditions, each of the form

$$\alpha\Phi + \beta\frac{d\Phi}{dx} = 0 \tag{III.3}$$

for some constant values of α and β, which are imposed at the end points of an interval $a \leq x \leq b$. [Note: if $p(x) = 0$ at an end point, the corresponding appropriate end condition may require merely that $\Phi(x)$ remain *finite* at that point.]

The function f may be a given direct function of x, or it may also depend upon x indirectly by also involving the unknown function $\Phi(x)$, and so being expressible in the form

$$f(x) = F(x, \Phi(x)). \tag{III.4}$$

In order to obtain a convenient reformulation of this problem, the *Green's function* G must first be determined, for a given ξ, given by $G_1(x)$ when $x < \xi$ and by $G_2(x)$ when $x > \xi$, and which has the following four properties [Ref. 13]:

1. The functions G_1 and G_2 satisfy the equation $\mathcal{L}\, G = 0$ in their intervals of definition, that is, $\mathcal{L}\, G_1 = 0$ when $x < \xi$, and $\mathcal{L}\, G_2 = 0$ when $x > \xi$.

2. The function G satisfies the homogeneous conditions prescribed at the end points $x = a$ and $x = b$; that is, G_1 satisfies the condition prescribed at $x = a$, and G_2 that corresponding to $x = b$.

3. The function G is continuous at $x = \xi$; that is, $G_1(\xi) = G_2(\xi)$.

4. The derivative of G has a discontinuity of magnitude $-1/p(\xi)$ at the point $x = \xi$; that is, $G_2'(\xi) - G_1'(\xi) = -1/p(\xi)$.

It is assumed that the function $p(x)$ is continuous and nonzero inside the interval (a, b), so that the discontinuity in the derivative of G is of finite magnitude, and also that $p'(x)$ and $q(x)$ are continuous in (a, b).

Then, when the function $G(x, \xi)$ exists, the original formulation of the problem can be written as

$$\Phi(x) = \int_a^b G(x, \xi)\ f(\xi)\ d\xi, \tag{III.5}$$

in the sense that this equation defines the *solution* of the problem when f is a given direct function of x, whereas the equation constitutes an equivalent *integral equation* problem when f involves Φ.

When the prescribed boundary conditions are not homogeneous, a modified procedure is needed. In this case, denote by $G(x, \xi)$ the Green's function corresponding to the associated *homogeneous* boundary conditions, and attempt to determine a function $P(x)$ such that the relation

$$\Phi(x) = P(x) + \int_a^b G(x, \xi)\ f(\xi)\ d\xi \tag{III.6}$$

is equivalent to the differential equation

$$\mathcal{L}\ \Phi(x) + f(x) = 0, \tag{III.7}$$

together with the prescribed *nonhomogeneous* boundary conditions. Since

$$\mathcal{L} \int_a^b G(x, \xi)\ f(\xi)\ d\xi\ =\ -f(x), \tag{III.8}$$

the differential equation takes the form $\mathcal{L}\ P(x) = 0$ and, since the Green's function integral satisfies the associated homogeneous bondary conditions, it follows that the function $P(x)$ must be the solution to the above-stated homogeneous equation with the prescribed nonhomogeneous boundary conditions. The existence of $P(x)$ is insured when $G(x, \xi)$ itself exists.

The Green's function approach can be expanded to higher dimensions, as in this research. The usefulness of a Green's function solution rests on the fact that the Green's function is independent of the nonhomogeneous term in the partial differential equation. Thus, once the Green's function is determined, the solution of the boundary value problem for different nonhomogeneous terms f(x) is obtained by a single integration.

In this research, the nonhomogeneous term is only on the boundary. Note that flow equations that are considered separately from thermal equations are homogeneous with homogeneous boundary conditions, except along the surface, where the thermal gradient gives rise to nonhomogeneous boundary conditions. By defining a Green's function in two dimensions as $G(x, y, \xi)$, the solution $\Phi(x, y)$ can be found as $\Phi(x, y) = \int_a^b G(x, y, \xi)\ f(\xi)\ d\xi$. Note further that no determination of arbitrary constants is required, since $\Phi(x, y)$ as given by the Green's function integral formula automatically satisfies the boundary conditions, as stated earlier.

The Green's function $G(x, y, \xi)$ can be identified as the response at the point ξ to a forcing function f which represents a unit impulse at the point x, with homogeneous boundary conditions. A more general nonhomogeneous term f on an interval $a < x < b$ can be regarded as a set of impulses with $f(x)$ giving the magnitude of the impulse at the point x. The solution of a nonhomogeneous boundary value problem in terms of a Green's function integral can then be interpreted as the result of su-

perimposing the responses to the set of impulses represented by the nonhomogeneous forcing term $f(x)$.

B. GREEN'S FUNCTION FOR FLOW

The flow field can be represented using the Green's function for a point force near a rigid wall, directed toward the wall. The Green's function for the stream function due to a unit (dimensionless) point force in the x direction applied on the surface at the point (ξ,0) is:

$$G(x,y,\xi) \equiv \frac{1}{2\pi}\left[y \log\left(\frac{r_+}{r_-}\right) - \frac{2xy\xi}{r_+^2}\right] \tag{III.9}$$

where

$$r_+ \equiv \sqrt{(x+\xi)^2+y^2}, \quad r_- \equiv \sqrt{(x-\xi)^2+y^2} \tag{III.10}$$

Blake [Ref. 14] and Hasimoto and Sano [Ref. 15] provide a detailed derivation of the form of the Green's function used herein. Treating the effect of the wall in two dimensions only, one can derive the Lorentz formula for the image system due to the presence of the wall. The image system consists of a combination of a Stokeslet, Stokes-doublet, and source-doublet, and is used to satisfy the no-slip condition at the wall. A flowfield plot of the Green's function with $\xi = 1$ is shown in Figure 3. Notice the behavior of the point force at $x = 1$. The flow field is concentrated at the point force and spreads out in the form of an ellipse from the cold corner. As $r \gg \xi$, the velocity decays rapidly. Also, more importantly, the flow down the vertical wall moves *away* from the wall (y axis), except near the surface where $y = 0$.

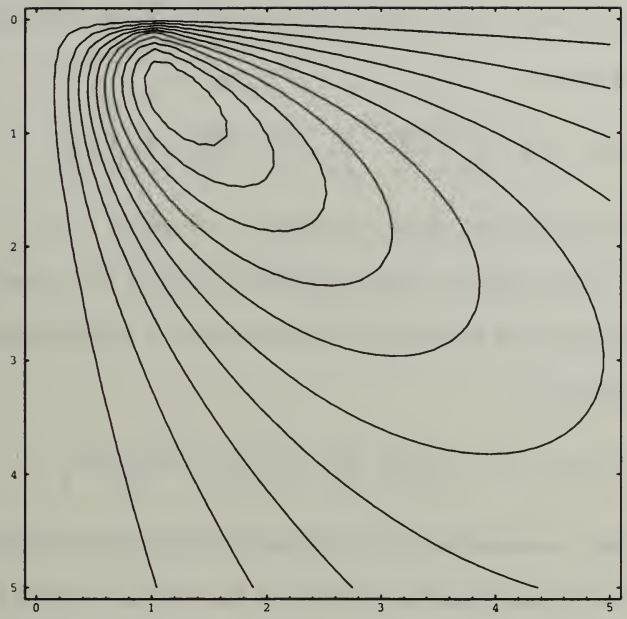

Figure 3. Green's Function with Point Force at x = 1: Streamlines (contours of stream function) are shown for a portion of the domain. Closely spaced streamlines indicate high velocities near the point force. Flow is in the counter-clockwise direction.

1. Velocity Components for Point Force

The velocity components of the stream function are designated as *uderiv* and *vderiv*, where *uderiv* is the partial derivative of the Green's function with respect to the y-direction, and *vderiv* is the negative of the partial derivative of the Green's function with respect to the x-direction as stated earlier. A simplified expression for *uderiv* is given by:

$$uderiv = G_y = \frac{1}{2\pi}\left[\frac{-2\xi x}{r_+^2} - \frac{y^2}{r_-^2} + \frac{y^2}{r_+^2} + \frac{4\xi x y^2}{r_+^4} + \log\left(\frac{r_+}{r_-}\right)\right] \tag{III.11}$$

A short analysis of *uderiv* shows a singularity of type $\log\frac{1}{r_-}$. As $r_- \to 0$, $y \to 0$ and $\xi \to x$. This relates to an infinite theoretical value of u, the horizontal velocity component, which will be discussed in the next section. A simplified expression for *vderiv* is given by:

$$vderiv = -G_x = \frac{1}{2\pi}\left[\frac{\xi y}{r_-^2} - \frac{\xi y}{r_+^2} - \frac{xy}{r_-^2} + \frac{xy}{r_+^2} + \frac{4\xi x(\xi + x)y}{r_+^4}\right] \tag{III.12}$$

Alternatively, we can determine the *uderiv* and *vderiv* components by taking vectors tangent to the Green's function streamlines at any point and breaking them down into horizontal (*uderiv*) and vertical (*vderiv*) components.

Consider a point force at the point $\xi = 1$ unit from the corner (in the x-direction). As stated earlier, *uderiv* and *vderiv* are determined for the point force near the corner, and a velocity vector field plot of horizontal versus vertical velocity components can be obtained. To gain a better understanding, first consider a large-scale view, as both the x- and y-directions vary from the corner (0,0) (top left of

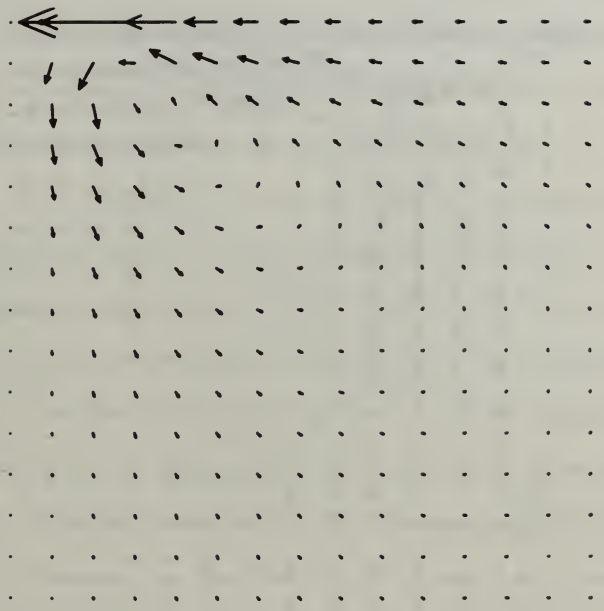

Figure 4. Velocity Vector Field of Point Force at x = 1: The direction and magnitude of the velocity in the domain ($0 \leq x \leq 5$, $0 \leq y \leq 5$) are denoted by the arrowhead on each line segment. The magnitude is greatest at the point force, directed toward the cold wall.

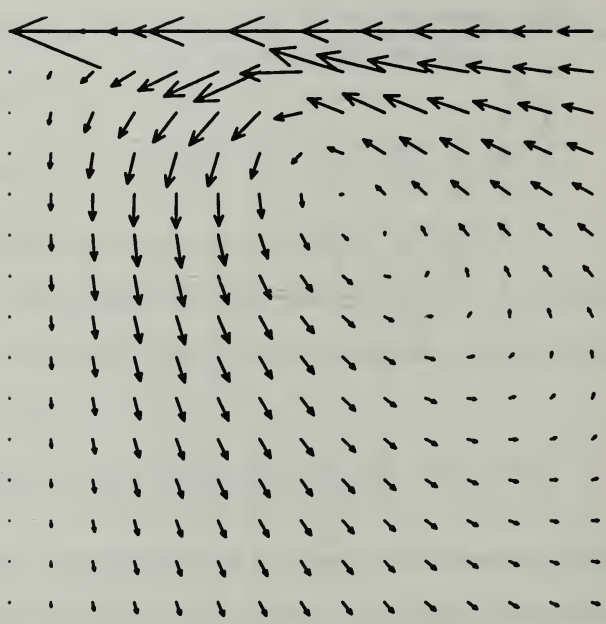

Figure 5. Magnified Velocity Vector Field of Point Force at x = 1: This view, with domain $0 \leq x \leq 2$, $0 \leq y \leq 2$, further emphasizes the greater surface velocity at the point force. The flow down from the surface is away from the wall.

plot) to a distance "far" away (5,5) (bottom right of plot), as shown in Figure 4. The longer the line segment, the greater the magnitude of the velocity. The direction of the velocity is denoted by an arrowhead on the line segment. Notice that the velocity has a larger magnitude at the point force than deep into the fluid. At the surface ($y = 0$), the theoretical value from (III.11) of the horizontal component of the velocity, u, at the point force is infinite; the vertical component $v = 0$ along the entire surface, as determined from (III.12), as well as along the wall.

Now, "magnify" the picture, with the far corner (bottom, right) coordinates at (2,2) (see Figure 5). The arrow scaling factor is the same as in Figure 4, to give perspective. Notice that the counter-clockwise flow with large magnitudes at the point force (0,1). There appears to be a noticeable increase in magnitude at the point force. As the flow moves down from the free surface, notice again that the flow is *away* from the wall, and begins to trace out an approximately elliptic streamline.

2. Velocity Components for Thermocapillary Forcing

Thermocapillary forces deal with the variation of surface tension with temperature along the surface of the domain. The velocity components of the thermocapillary forcing are the nonhomogeneous terms that sum up the changes in temperature times the partial derivatives of the Green's function (*uderiv* and *vderiv*) over the length of the surface. In particular, $T'(\xi)$ is the function used to weight the point sources along the free surface.

In abbreviated form, the horizontal component of the velocity can be calculated as:

$$u(x,y) = \int_0^\infty T'(\xi)\ uderiv(x,y,\xi)\ d\xi \qquad \text{(III.13)}$$

Similarly, the vertical component of the velocity appears as:

$$v(x,y) = \int_0^\infty T'(\xi)\ vderiv(x,y,\xi)\ d\xi \qquad \text{(III.14)}$$

IV. CONDUCTIVE CASE: $M = 0$

To compare the convective effects of higher Marangoni number flows, the conductive case $M = 0$ is used as a comparative base. With $M = 0$, no effects of convection exist. Thus an analytic solution for the temperature distribution can be obtained, which should be valid as an initial distribution for small M values.

A. ANALYTIC SOLUTION FOR TEMPERATURE

Since the normal derivative (T_y) on the boundary of the free surface is zero, the quarter-plane problem can be mirrored into a half-plane problem. The homogeneous Neumann condition is thus absorbed into the half-plane problem by symmetry. This gives a Dirichlet problem for the right half of the xy plane.

The method of solution is to obtain a new Dirichlet problem with conformal mapping for a region in the complex uv plane. Note that the u and v here are the components of the complex plane $w = u + iv$, not the same as the velocity components. Since a function which is harmonic in a simply connected domain always has a harmonic conjugate, the solution of this boundary value problem for such a domain is the real or imaginary part of the analytic function. That region will be the image of the half plane under a transformation $w = f(z)$ which is analytic in the domain $x > 0$ and which is conformal along the boundary $x = 0$ except at the points $(0, \pm 1)$ where it is undefined. Two theorems are now applied.

Theorem 1 (Transformations of Harmonic Functions.) *Suppose that an analytic function*

$$w = f(z) = u(x,y) + i\ v(x,y) \tag{IV.1}$$

maps a domain D_z in the z plane onto a domain D_w in the w plane. If $h(u,v)$ is a harmonic function defined on D_w, then the function

$$H(x,y) = h\,[u(x,y),\ v(x,y)] \tag{IV.2}$$

is harmonic in D_z.

Theorem 2 (Transformations of Boundary Conditions.) *Suppose that a transformation*

$$w = f(z) = u(x,y) + i\ v(x,y) \tag{IV.3}$$

is conformal on a smooth arc C, and let Γ be the image of C under that transformation. If, along Γ, a function $h(u,v)$ satisfies either of the conditions

$$h = h_0 \quad or \quad \frac{dh}{dn} = 0 \tag{IV.4}$$

where h_0 is a real constant and $\frac{dh}{dn}$ denotes derivatives normal to Γ, then, along C, the function

$$H(x,y) = h\,[u(x,y),\ v(x,y)] \tag{IV.5}$$

satisfies the corresponding condition

$$H = h_0 \quad or \quad \frac{dH}{dN} = 0 \tag{IV.6}$$

where $\frac{dH}{dN}$ denotes derivatives normal to C.

The proofs of these theorems can be found in *Complex Variables and Applications*, by Churchill and Brown [Ref. 16].

A harmonic function of u and v will be transformed into a harmonic function of x and y, and the boundary conditions in the uv plane will be preserved on corresponding portions of the boundary in the xy plane. Let us write:

$$z - i = r_1 e^{i\theta_1} \quad and \quad z + i = r_2 e^{i\theta_2} \tag{IV.7}$$

where $\pi \leq \theta_k < -\pi \quad (k = 1, 2)$.

The transformation $w = \log\left(\frac{z-i}{z+i}\right) = \ln\frac{r_1}{r_2} + i(\theta_1 - \theta_2)$ is defined on the right half plane $x \geq 0$ except at $z = \pm i$. The right half plane maps to the strip $-\pi < v < 0$ in the w plane. The cold wall ($T = -1$, the imaginary axis between $-i$ and i) maps to the line $v = \pi$ in the complex plane. The surface of the pool (with $T_y = 0$) maps to the real axis $v = 0$, with $u < 0$, and the rest of the wall (imaginary axis above i and below $-i$) maps to $v = 0$. A bounded harmonic function of u and v that satisfies the boundary conditions is $T = -\frac{1}{\pi}v$. It is harmonic since it is the imaginary part of the entire function $\left(-\frac{1}{\pi}w\right)$. If $v = 0$, $T = 0$; $v = \pi$, $T = -1$. Since $v = \mathcal{I}(w) = \arg(\frac{z-i}{z+i})$, we solve for v:

$$v = \arg\left[\frac{x + (y-1)i}{x + (y+1)i}\right] \tag{IV.8}$$

$$= \arg\left[\frac{x^2 + y^2 - 1 - (2x)i}{x^2 + (y+1)^2}\right] \tag{IV.9}$$

$$= \arctan \frac{-2x}{x^2 + y^2 - 1} \tag{IV.10}$$

So, $T(x,y) = -\frac{1}{\pi}v$, which gives:

$$T(x,y) = \frac{1}{\pi} \arctan \frac{2x}{x^2 + y^2 - 1} \tag{IV.11}$$

Since the function $T = -\frac{1}{\pi}v$ is harmonic in the strip $-\pi < v < 0$, and the transformation is analytic in the half plane $x > 0$, Theorem 1 is applied to conclude that the function $T(x,y) = \frac{1}{\pi} \arctan \frac{2x}{x^2+y^2-1}$ is harmonic in that half plane. The boundary conditions for the two harmonic functions are the same in corresponding parts of the boundaries because they are the type $h = h_0$ (as required in Theorem 2). The bounded function $T(x,y)$, therefore, is the desired solution in the case where the Marangoni number M = 0. A 3-dimensional plot of this temperature field $T(x,y)$ is shown in Figure 6. Note the discontinuity down the vertical wall at the point (0, 1). The velocity components of the thermocapillary flow will depend on the derivative of the temperature solution with respect to the x-direction, which becomes:

$$T'(x) \equiv T_x(x,0) = \frac{-2}{\pi(1 + x^2)} \tag{IV.12}$$

A two-dimensional plot of this derivative is shown in Figure 7. As expected, $T'(x)$ dies out as x increases.

B. RESULTING FLOW

To solve for the velocity components of the thermocapillary flow (with $M = 0$), it is necessary to integrate numerically the product of the *uderiv* and *vderiv* components of the Green's function (as functions of x, y, and ξ) with $T'(\xi)$, from zero to

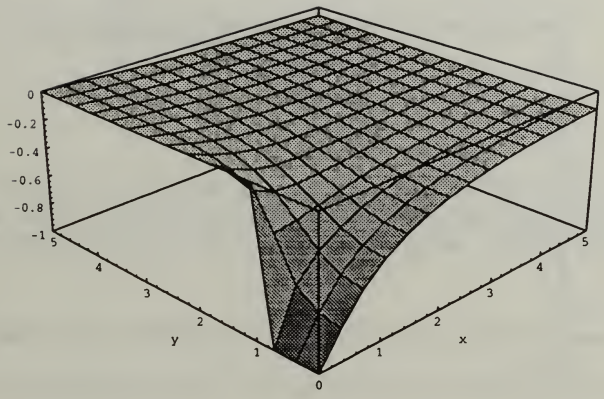

Figure 6. Initial Temperature Field $T(x,y) = \frac{1}{\pi} \arctan \frac{2x}{x^2+y^2-1}$: Three-dimensional plot of harmonic temperature field. The wall (y-axis) is maintained at a cold temperature to a depth of unity, with warmer temperature below.

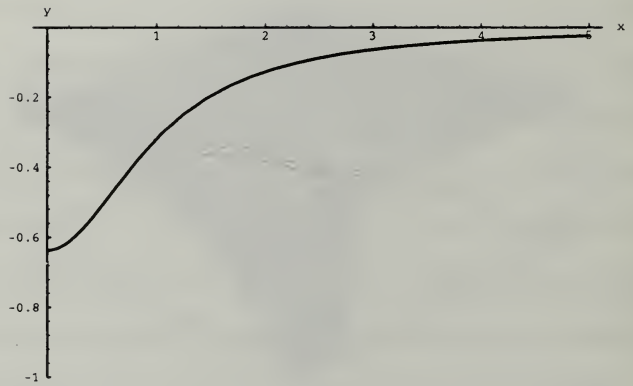

Figure 7. $T'(x,0) = \frac{-2}{\pi(1+x^2)}$: Plot of the derivative of the temperature solution. As x increases, $T_x(x,0)$ dies out. This derivative is used in the calculation of the velocity components.

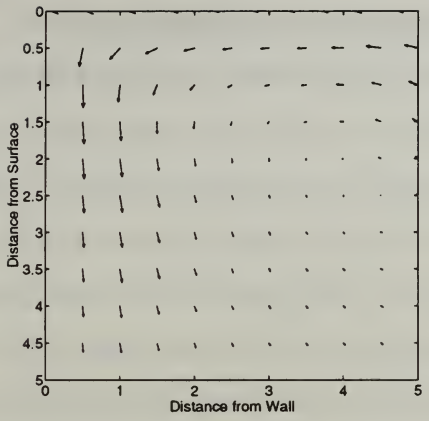

Figure 8. Velocity Field Using Numerical Integration ($M = 0$): Obtained by integrating the product of *uderiv* and *vderiv* components of the Green's function with $T'(\xi)$. The velocity decays as it flows down from the surface and away from the wall.

infinity. *Mathematica's* **NIntegrate** function was used to perform numerical integration in computing the values of $u(x, y)$ and $v(x, y)$. This function attempts to numerically integrate over a range that includes singularities at intermediate points x_i. If there are no singularities, the result is equivalent to an integral from 0 to ∞.

Finally, results from the numerical integration are combined into a vector field plot. Figure 8 shows the velocity field from the corner out to a distance of 5 units from the wall and surface. The velocity is extremely rapid at the surface, as expected, and the velocity is driven to the corner in a counter-clockwise direction (i.e., the flow all along the surface is toward the wall). The flow is then down from the surface along

the wall, into the bulk fluid. As the fluid flows down from the surface, the velocity decays rapidly and it also flows away from the wall as it decays. This rapid decay of velocity implies that *no* thermal boundary layer forms on the wall.

V. NUMERICAL METHODS

To model the revised energy equation (II.9), two numerical methods were explored. The first, a *FORTRAN* subroutine from the IMSL, Inc. library, treated the right-hand-side of (II.9) as an input function and solved the remaining Poisson equation. The second method involved solving (II.9) by the standard alternating-direction implicit (ADI) method, using *MATLAB*, a high-performance interactive software package for numerical computation. To use these two approaches, the domain had to be modified to be a finite one. (Both numerical schemes required that a finite domain be used). After some initial calculations, it was determined that the domain should be a rectangle with two sides as the surface and wall containing the cold corner, and the other two sides a distance of four units in the horizontal (parallel to the surface) direction from the wall, and three units in the vertical direction from the surface. Recall that the wall is kept to a constant temperature only to a depth of one unit. Far away from the cold corner, the fluid temperature is warmer than at the corner. The boundary conditions as $x,\ y \to \infty$ are $T \to 0$; thus $T = 0$ at $x = 4$ and $y = 3$. Results revealed that the size of the rectangular domain accurately modeled the much more complex real geometry of the problem.

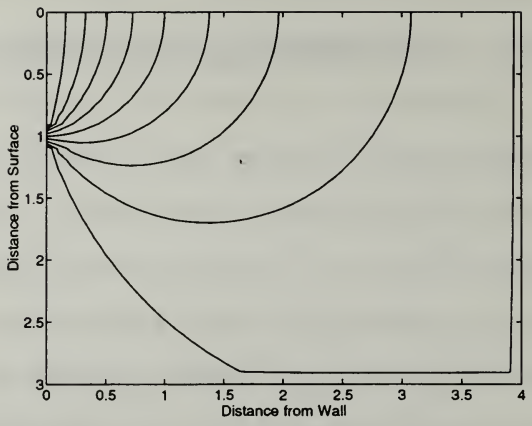

Figure 9. Conduction Solution of the Temperature, $T(x, y)$: Contour plot of initial distribution used for numerical methods. The homogeneous boundary conditions at the ends of the domain ($x = 4$, and $y = 3$) give rise to the shape of the outer contour line (isotherm).

A. POISSON APPROXIMATION (*FORTRAN*)

As a first approach to solve (II.9), existing algorithms were investigated and a *FORTRAN* routine was selected. It has the flexibility of modeling the problem without many complications. The only drawback was finding a convenient method to graphically display the output data. The conduction solution of the temperature distribution, $T(x, y) = \frac{1}{\pi} \arctan \frac{2x}{x^2+y^2-1}$, is shown in Figure 9. Using *Mathematica*, values for M $\mathbf{u} \cdot \nabla T$ were calculated in a rectangle, with $\Delta x = \Delta y$ on a grid of $0 \leq x \leq 4$ by $0 \leq y \leq 3$. A table of values was generated using a given grid size and M = 1. These values were then read into a *FORTRAN* program to solve the Poisson

equation:

$$\nabla^2 T = f(x,y) \tag{V.1}$$

using $f(x,y) = M\ \mathbf{u} \cdot \nabla T$ from the input file. The tables of $\mathbf{u} \cdot \nabla T$ were generated for various grid sizes. A modification of the *FORTRAN* subroutine FPS2H from the IMSL, Inc. library [Ref. 17, 18] was employed. The algorithm solves the equation

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + cT = p \tag{V.2}$$

on a rectangular domain (as specified above). In this case, $c = 0$ and p is the input file $f(x,y)$. By setting the right-hand-side of the Poisson equation equal to $f(x,y)$, an approximation to the convection-diffusion equation is obtained. Note that the *FORTRAN* program solves the Poisson equation, not the convection-diffusion equation. The forcing function term in $f(x,y)$ in the Poisson equation should depend on the solution temperature T, as it does in the convection-diffusion equation.

A copy of the subroutine is attached in Appendix A. The input arguments are the *Mathematica*-generated value for each (x,y) grid point (a user-supplied function to evaluate the right side of the partial differential equation) and the user-supplied boundary conditions, as well as the grid sizes (in both x- and y- directions), and the output is the solution to the Poisson equation. The routine solves Poisson's Equation on a two-dimensional rectangle using a fast Poisson solver based on a finite difference scheme on a uniform mesh. It discretizes the problem and then solves it using Fourier transforms [Ref. 18].

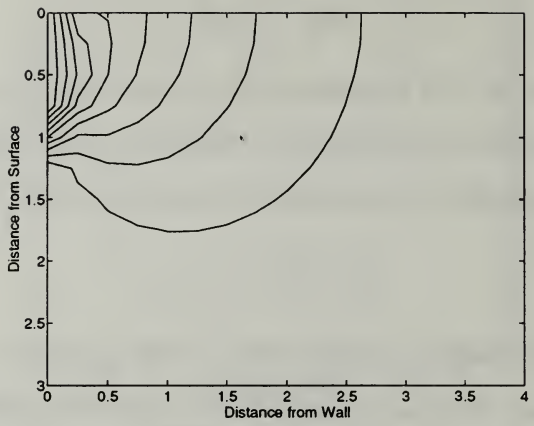

Figure 10. Temperature Distribution obtained from *FORTRAN* Routine ($h = 0.25$, $M = 0$): Contour plot of temperature distribution (purely conductive).

Figure 10 depicts the purely conductive temperature distribution ($M = 0$ case) obtained from the *FORTRAN* subroutine, using a uniform mesh of 0.25. Figure 11 illustrates the *FORTRAN* temperature solution when the Marangoni number is increased to $M = 1$. Notice the conductive behavior of the flow field. The flow has a counter-clockwise motion, directed toward the cold wall near the surface and then away from the wall as it flows away from the surface. Figure 11 shows virtually no change from the $M = 0$ case. With the *FORTRAN* approach, even the case where $M = 10$ showed no substantial change from Figure 10. However, as M is increased, the resulting temperature solution becomes less stable. For example, when $M = 100$, the numerical method broke down and unstable results were obtained.

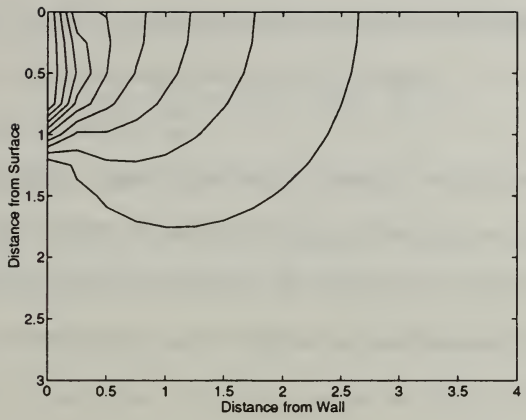

Figure 11. Temperature Distribution obtained from *FORTRAN* Routine ($h = 0.25$, $M = 1$): Contour plot of temperature solution showing no evidence of convection in the flow.

As mentioned above, this approach had limitations. To compute the input data with *Mathematica* was very time- and resource-consuming, due to the numerical integration algorithm in *Mathematica*. Also, output data had to be transferred to and rewritten as *Mathematica* files, in order to graphically display the results. Moreover, the Poisson approximation is only valid for small M values. Thus, the effort for one time-step iteration was somewhat complicated, and an alternate numerical approach was investigated.

B. FULL EQUATION (*MATLAB*)

The time-dependent problem can be expanded as:

$$M\left(\frac{\partial T}{\partial t}+u\frac{\partial T}{\partial x}+v\frac{\partial T}{\partial y}\right)=\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2} \tag{V.3}$$

It is assumed that the time-dependent solution will converge to a steady-state solution. Equation (V.3) was solved using an alternating direction implicit (ADI) method as developed by Polezhaev in 1967 [Ref. 19]. This method treats each time step as two half steps. The above equation can be substituted into the scheme as:

STEP 1:

$$\left[1+\frac{\Delta t}{2}\left(u_{i,j}\frac{\bar{\delta}_x}{2\,\Delta x}-\frac{1}{M}\hat{\delta}_x^2\right)\right]T_{i,j}^{*}=\left[1-\frac{\Delta t}{2}\left(v_{i,j}\frac{\bar{\delta}_y}{2\,\Delta y}-\frac{1}{M}\hat{\delta}_y^2\right)\right]T_{i,j}^{n} \tag{V.4}$$

STEP 2:

$$\left[1+\frac{\Delta t}{2}\left(v_{i,j}\frac{\bar{\delta}_y}{2\,\Delta y}-\frac{1}{M}\hat{\delta}_y^2\right)\right]T_{i,j}^{n+1}=\left[1-\frac{\Delta t}{2}\left(u_{i,j}\frac{\bar{\delta}_x}{2\,\Delta x}-\frac{1}{M}\hat{\delta}_x^2\right)\right]T_{i,j}^{*} \tag{V.5}$$

where the central-difference operators $\bar{\delta}$ and $\hat{\delta}^2$ are defined as:

$$\begin{aligned}\bar{\delta}_x\,T_{i,j} &= T_{i+1,j}-T_{i-1,j}\\ \bar{\delta}_y\,T_{i,j} &= T_{i,j+1}-T_{i,j-1}\end{aligned} \tag{V.6}$$

and

$$\begin{aligned}\hat{\delta}_x^2\,T_{i,j}^{n} &= \frac{T_{i+1,j}^{n}-2\,T_{i,j}^{n}+T_{i-1,j}^{n}}{(\Delta x)^2}\\ \hat{\delta}_y^2\,T_{i,j}^{n} &= \frac{T_{i,j+1}^{n}-2\,T_{i,j}^{n}+T_{i,j-1}^{n}}{(\Delta y)^2}\end{aligned} \tag{V.7}$$

As a result of splitting the time step in this algorithm, only tridiagonal systems of linear algebraic equations must be solved (at each step). This method is first-order accurate with a truncation error of $O[\Delta t,\ (\Delta x)^2,\ (\Delta y)^2]$ and is unconditionally stable for the linear (no convection) case.

The scheme was coded using MATLAB 4.0 with $\Delta x = \Delta y$ on a rectangular grid of $0 \leq x \leq 4$ by $0 \leq y \leq 3$. The problem's boundary conditions as stated earlier were incorporated into the scheme. In addition, the temperature at the extreme boundaries ($x = 4$ and $y = 3$) was set to zero. A copy of the code is attached in Appendix B. As noted in the comment lines of the code, to set the boundary conditions at each of the four sides of the rectangle, some of the coefficients for $T^n_{i,j}$, $T^*_{i,j}$, and $T^{n+1}_{i,j}$ had to be manipulated. By setting these coefficients to zero at key positions, the required boundary conditions for the iteration were preserved. These manipulations ensured adherence to the boundary conditions without altering the solution method inside the boundaries.

Several subroutines are called from the main program, ADI.m. These include: TINIT.m, TRAP.m, UDERIV.m, VDERIV.m, CROUT.m, and CROUTSLV.m. The first of these subroutines, TINIT.m, creates the initial temperature distribution based on the chosen grid size:

$$T(x,y) = \frac{1}{\pi} \arctan \frac{2x}{x^2 + y^2 - 1} \tag{V.8}$$

with appropriate boundary conditions. To perform the numerical integration in (2), TRAP.m uses a trapezoidal technique in summing the integrand for all ξ between 0

and the far right edge ($x = 4$). UDERIV.m and VDERIV.m calculate the value of the respective derivative to the Green's Function at the particular (x, y) point in the grid.

The calculation of *vderiv* is straight-forward; however, *uderiv* has a logarithmic singularity along the surface ($y = 0$) at $x = \xi$. Thus, the algorithm had to be modified, in order to solve for the effective *uderiv* value at the point of singularity. Let this value be u_{peak}, and let the grid spacing be denoted as h. Using a trapezoidal representation around the singularity, an expression Δu is obtained for the contribution of this region:

$$\Delta u(x,0) = \int_{\xi-h}^{\xi+h} T'(x)\ uderiv(x,0)\ dx \tag{V.9}$$

or, by the trapezoidal rule,

$$\Delta u(x,0) \simeq\ T'(\xi)\ h\left[\frac{1}{2}\ uderiv(\xi-h,0) + u_{peak} + \frac{1}{2}\ uderiv(\xi+h,0)\right] \tag{V.10}$$

assuming that T' does not change drastically in the vicinity of the singularity. Recall that *uderiv* (III.11) is defined as:

$$uderiv = \frac{1}{2\pi}\left[\frac{-2\xi x}{r_+^2} - \frac{y^2}{r_-^2} + \frac{y^2}{r_+^2} + \frac{4\xi x y^2}{r_+^4} + \log\left(\frac{r_+}{r_-}\right)\right] \tag{V.11}$$

Thus, setting $y = 0$ gives:

$$uderiv(x,0) = \frac{1}{2\pi}\left[\log(x+\xi) - \log|x-\xi| - \frac{2\xi x}{(x+\xi)^2}\right] \tag{V.12}$$

Letting $u'(x,0) = (2\pi)\ \Delta u(x,0)/T'(\xi)$ yields:

$$u'(x,0) \;=\; \int_{\xi-h}^{\xi+h} \log(x+\xi)\ dx - \int_{\xi-h}^{\xi+h} \log|x-\xi|\ dx - \int_{\xi-h}^{\xi+h} \frac{2\xi x}{(x+\xi)^2} dx \tag{V.13}$$

$$= \int_{\xi-h}^{\xi+h} \log(x+\xi)\,dx - \int_{\xi-h}^{\xi} \log(\xi - x)\,dx - \int_{\xi}^{\xi+h} \log(x-\xi)\,dx$$
$$- \int_{\xi-h}^{\xi+h} \xi \left[\frac{2(x+\xi)}{(x+\xi)^2} - \frac{2\xi}{(x+\xi)^2}\right] dx \qquad \text{(V.14)}$$

$$= \int_{2\xi-h}^{2\xi+h} \log u\,du - \int_{0}^{h} \log u\,du - \int_{0}^{h} \log u\,du$$
$$- \int_{\xi-h}^{\xi+h} 2\xi \left[\frac{1}{x+\xi} - \frac{\xi}{(x+\xi)^2}\right] dx \qquad \text{(V.15)}$$

Evaluating these integrals produces:

$$u'(x,0) = 2\xi \log(2\xi+h) - (2\xi+h) + h\log(2\xi+h) - 2\xi\log(2\xi-h) + (2\xi-h)$$
$$+ h\log(2\xi-h) - 2h\log h + 2h - 2\xi\log(2\xi+h) - \frac{2\xi^2}{2\xi+h}$$
$$+ 2\xi\log(2\xi-h) + \frac{2\xi^2}{2\xi-h} \qquad \text{(V.16)}$$

$$= h\left[\log(2\xi+h) + \log(2\xi-h) - 2\log h + \frac{4\xi^2}{4\xi^2-h^2}\right] \qquad \text{(V.17)}$$

so that

$$\Delta u(x,0) = T'(\xi)\,\frac{h}{2\pi}\left[\log(2\xi+h) + \log(2\xi-h) - 2\log h + \frac{4\xi^2}{4\xi^2-h^2}\right] \qquad \text{(V.18)}$$

Setting this equal to the trapezoidal representation and dividing both sides by $T'(\xi)$ gives:

$$h\left[\frac{1}{2}\,uderiv(\xi-h,0) + u_{peak} + \frac{1}{2}\,uderiv(\xi+h,0)\right] =$$
$$\frac{h}{2\pi}\left[\log(2\xi+h) + \log(2\xi-h) - 2\log h + \frac{4\xi^2}{4\xi^2-h^2}\right] \qquad \text{(V.19)}$$

Expanding the trapezoidal term and multiplying through by $\frac{2\pi}{h}$ gives:

$$\frac{1}{2}\left[\log(2\xi - h) - \log(-h) - \frac{2\xi(\xi - h)}{(2\xi - h)^2}\right] + 2\pi u_{peak} + \frac{1}{2}\left[\log(2\xi + h) - \log(h) - \frac{2\xi(\xi + h)}{(2\xi + h)^2}\right] = \left[\log(2\xi + h) + \log(2\xi - h) - 2\log h + \frac{4\xi^2}{4\xi^2 - h^2}\right] \quad \text{(V.20)}$$

Simplifying:

$$2\pi u_{peak} = \frac{1}{2}\left[\log(2\xi + h) + \log(2\xi - h)\right] - \log h + \frac{4\xi^2}{4\xi^2 - h^2} + \frac{\xi^2 - \xi h}{(2\xi - h)^2} + \frac{\xi^2 + \xi h}{(2\xi + h)^2} \quad \text{(V.21)}$$

$$u_{peak} = \frac{1}{2\pi}\left[\frac{1}{2}\log\left(\frac{4\xi^2 - h^2}{h^2}\right) + \frac{2\xi^2(12\xi^2 - 5h^2)}{(4\xi^2 - h^2)^2}\right] \quad \text{(V.22)}$$

The last two subroutines, CROUT.m and CROUTSLV.m, solve the tridiagonal set of equations for the unknown left-hand-side of each equation in STEP 1 and STEP 2 of the ADI method. CROUT.m computes the LU factorization of a tridiagonal matrix. CROUTSLV.m then takes those two matrices, L and U, and solves the matrix equation $L\ U\ T_{new} = B$, by solving first $L\ y = B$, and then solving $U\ T_{new} = y$. In this case B is the matrix consisting of the right-hand-side of each equation in STEP 1 and STEP 2.

As the Marangoni number M increased, the steady-state results from earlier (lower M values) iterations were used as initial conditions. For example, the steady-state temperature and velocity values for the case where $M = 30$ were used as input readings for the $M = 100$ case. ADI.m was adjusted to read the steady-state values as input and begin iterating with those values. In this way, TINIT.m was not used, and

the steady-state temperature and velocity distributions were reached more efficiently. Since the grid spacing, h, for $M = 30$ was larger than that used for $M = 100$ (as will be explained in the next chapter), linear grid interpolation of the temperature and velocity distributions was used. This was accomplished by stretching those matrices linearly with simple *MATLAB* matrix manipulation commands, creating 151 by 201 matrices from 31 by 41 matrices.

The MATLAB program solves the convection-diffusion equation much faster than *Mathematica* could. However, due to the nature of calculating initial temperatures and velocities at each grid point, several *for* loops are incorporated into the scheme, reducing *MATLAB*'s inherent efficiency.

VI. RESULTS

Solutions were calculated for several values of M, from $M = 1$ to $M = 300$. Increments were chosen to be roughly equal in a logarithmic sense.

A. M = 1

Small values of M, in this case $M = 1$, are characteristic of diffusion dominance. Recall that Figure 9 shows the initial temperature distribution over the domain for $M = 1$. The numerical scheme calculated temperature and velocity val-

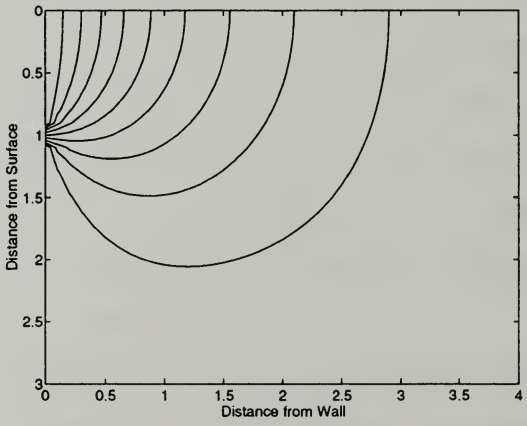

Figure 12. Steady-State Temperature Distribution from *MATLAB* routine ($h = 0.1$, $M = 1$): Contour plot of temperature solution showing no evidence of convection in the flow.

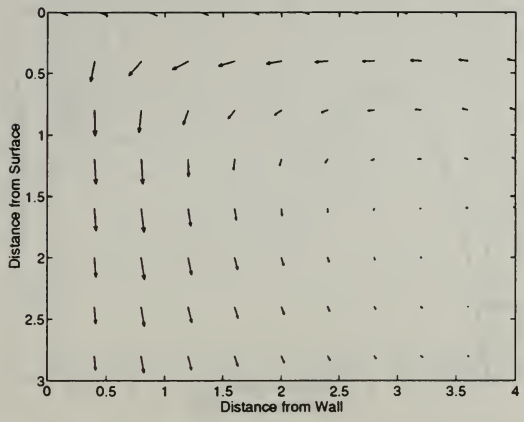

Figure 13. Steady-State Velocity Vector Field Profile ($h = 0.1$, $M = 1$): The velocity is highest at the surface, directed toward the wall, and then turns away from the surface along the wall.

ues to approximate steady-state conditions, in this case reached after 100 time steps. To satisfy the ADI scheme's stability requirement, a Courant number $\nu = 0.1$ was used. When a higher ν value was used at higher M values (for example, $\nu = 0.4$), unstable results were obtained after only a short number of iterations. Therefore, the value of $\nu = 0.1$ was used for all cases. Figure 12 shows the steady-state temperature distribution ($h = 0.1$, $M = 1$). There is no evidence of convection (as expected). When compared with Figure 10, which was obtained with the *FORTRAN* approach, there is no real difference. This validates that the initial temperature distribution was a reasonable starting point. Next, consider the velocity profile of the domain, as

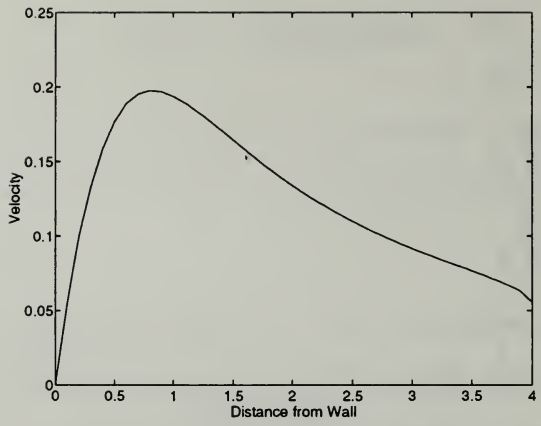

Figure 14. Steady-State Surface Velocity Profile ($h = 0.1$, $M = 1$): At approximately one unit from the wall, the velocity reaches maximum value, then drops sharply as the flow approaches the wall.

shown in Figure 13. The velocity is highest at the surface, directed toward the wall, and then turns away from the surface along the wall. At approximately one unit away from the wall (along the surface), the velocity reaches its maximum value (as seen in Figure 14), and it then drops sharply as it approaches the wall, also as expected.

B. M = 3

The Marangoni number was increased slightly from $M = 1$ to $M = 3$, keeping the uniform grid spacing $h = 0.1$. Figure 15 shows the steady-state temperature distribution of the domain. This figure is virtually identical to Figure 12 (where $M = 1$), implying no effects of convection in the distribution. Also, the surface temperature

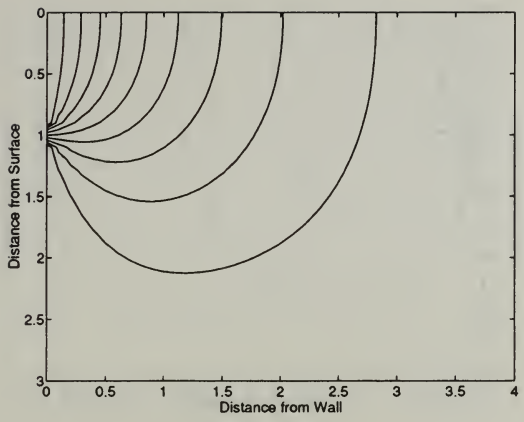

Figure 15. Steady-State Temperature Distribution ($h = 0.1$, $M = 3$): Contour plot of temperature solution (almost identical to $M = 1$ case) showing no evidence of convection in the flow.

profile, surface velocity profile and domain velocity profile are all virtually unchanged from the $M = 1$ case.

C. M = 10

As M increases, the temperature distribution gradually changes from a diffusion dominant distribution to one charateristic of high convection. With the case of $M = 10$, convection becomes slightly apparent. Figure 16 shows the steady-state temperature distribution ($h = 0.1$, $M = 10$). The temperature isotherms are pulled somewhat closer to the wall and the velocity components along the surface are slightly

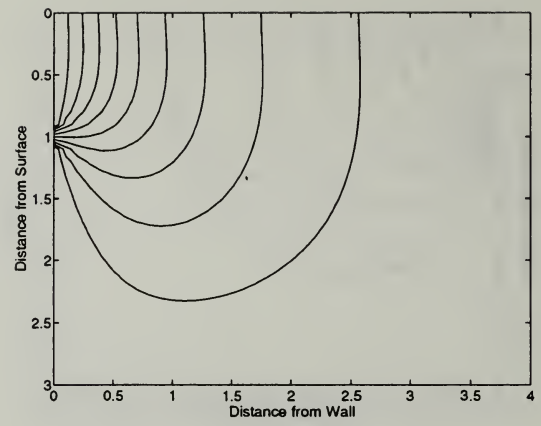

Figure 16. Steady-State Temperature Distribution ($h = 0.1$, $M = 10$): Contour plot of temperature solution shows slight evidence of convection in the flow.

higher than when $M = 1$ (see Figure 14). The temperature distribution is beginning the transition to convective behavior.

D. M = 30

It appears that the steady-state temperature distribution with $M = 30$ shows stronger effects of convective behavior, as illustrated in Figure 17. The temperature isotherms are drawing into the cold corner, leaving almost half of the problem domain at constant temperature (close to T = 0). The surface velocity has a maximum closer to the wall than in previous cases. Also, as will be discussed later, the surface temperature drops more drastically in a short distance from the wall, eventually leveling out in the bulk pool.

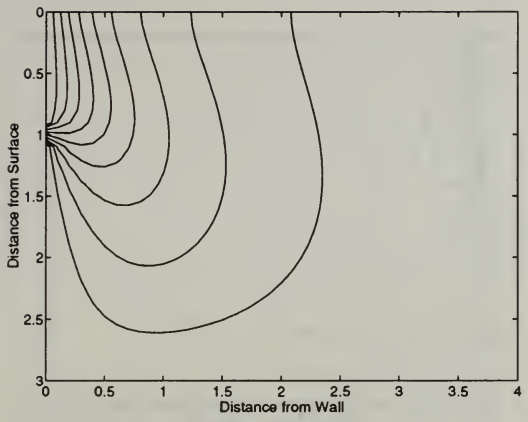

Figure 17. Steady-State Temperature Distribution ($h = 0.1$, $M = 30$): Contour plot of temperature solution shows stronger effects of convection in the flow. Closely spaced isotherms indicate rapid growth in temperature near the wall.

E. M = 100

The identical grid spacing $\Delta x = \Delta y = h = 0.1$ used with the cases $M = 1$, 3, 10, and 30 was employed with $M = 100$. This violated the stability requirement, resulting in numerically unstable results. The time step was then reduced to keep the grid spacing at $h = 0.1$. This resulted in a stable solution; however, resolution of the velocity profile close to the cold corner ($x \to 0$, $y \to 0$) was not sufficient to model what is actually taking place. Hence, a finer grid was required, and $h = 0.02$ was employed. Due to the amount of calculations per time step at each grid point, the numerical scheme's running time increased tremendously. To offset this factor, the algorithm was modified to calculate new velocities at every 10^{th} time step, instead

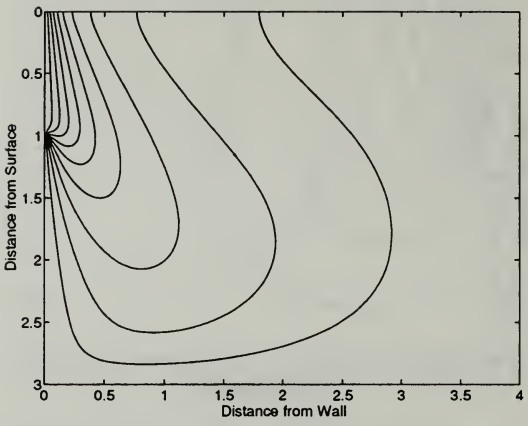

Figure 18. Steady-State Temperature Distribution ($h = 0.02$, $M = 100$): Contour plot of the temperature solution. The uniform grid spacing is reduced to 0.02 and the isotherms are very close to each other, forming a compressed region close to the wall.

of at every step, as in the cases of lower Marangoni numbers. Since the velocities at each point in the flow changed slightly (if at all) from time step to time step, this modification was deemed reasonable. In other words, new temperature values calculated with the ADI process used the same velocity values for ten consecutive time steps; the new velocities were then calculated at each grid point and the process was repeated until a steady-state temperature distribution was reached.

Figure 18 shows the steady-state temperature distribution for the case where $M = 100$, with a uniform grid spacing $h = 0.02$. The isotherms are very close to each other near the cold corner, indicating that a compressed region is forming there.

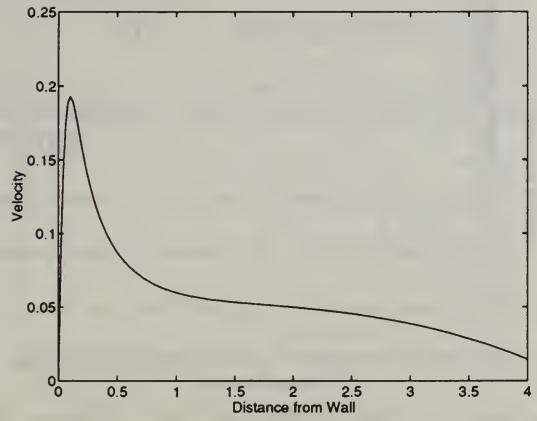

Figure 19. Steady-State Surface Velocity Profile ($h = 0.02$, $M = 100$): The surface velocity has a maximum closer to the wall than for lower M values, indicating a rapid decrease in velocity close to the cold corner.

Figure 19 shows the surface velocity plot of this case, with the velocity slowing significantly a few grid points from the wall ("beyond" the compressed region). However, the thermocapillary forcing by the velocity is limited to the region close to the wall, concentrated near the wall, so that no thermal boundary layer does form, as noted in [Ref. 20].

F. M = 300

The steady-state temperature distribution for the case of $M = 300$ is shown in Figure 20. As with the case of $M = 100$, the grid spacing had to be decreased (to $h = 0.005$) to offer sufficient resolution to study the surface velocity profile. It

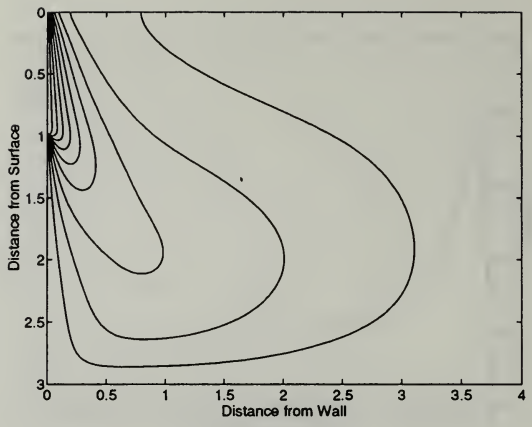

Figure 20. Steady-State Temperature Distribution ($h = 0.02$, $M = 300$): Contour plot of the temperature solution. The isotherms appear to meet at the wall-surface corner. The transition to convective behaviour is complete.

appears as if the first four isotherms meet in the wall-surface corner. The isotherms are almost on top of each other, exhibiting the above-mentioned compressed region in the flow.

The need for reduced grid spacing brought significant computational problems. A finer grid means more grid points, requiring separate temperature and velocity calculations at each point, requiring more computer storage space. With matrix sizes greater than 151 by 201, MATLAB's temporary data file storage capacity was filled to maximum, preventing a completion of the data run. For example, when, $h = 0.005$, a 481401 ($601 \cdot 801$) by 3 matrix used in the tridiagonal solver (CROUTSLV.m) requires over 10 Mbytes of storage alone.

VII. DISCUSSION

A. COMPARISON OF GRID SIZE VS MARANGONI NUMBER

The ADI scheme is unconditionally stable in the linear case [Ref. 19]. However, as the Marangoni number increased ($M \geq 100$), it was necessary to decrease the grid size h, to offer sufficient resolution of the surface velocity profile, while still maintaining numerically stable results. To effectively model the surface velocity, which will be explained in detail later, required several grid points in the regions of large variations (close to the wall in steady-state). Thus, a successful "rule of thumb" was to try to keep the grid size inversely proportional with the Marangoni number. With M equal to 1, 3, 10, and even 30, a grid size $h = 0.1$ was sufficient. As M increased to 100, h decreased to 0.02. As M increased to 300, h had to decrease to 0.005. However, as stated earlier in the previous chapter, the MATLAB program could not compute such large matrices. With $M = 1000$, a grid size of $h = 0.001$ is required.

B. COMPARISON OF SURFACE TEMPERATURE VS MARANGONI NUMBER

As the Marangoni number M increases, thermal convection becomes more dominant than thermal conduction. Figure 21 shows the steady-state surface temperature profiles for the various Marangoni values (1, 3, 10, 30, 100, and 300). Comparing

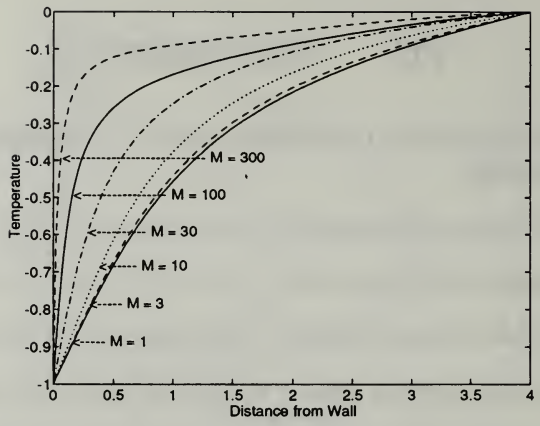

Figure 21. Steady-State Surface Temperature for Various M Values: This plot shows that as M increases, the surface temperature rises more rapidly as the distance from the wall increases. As the flow becomes more convective (high M), the change in surface temperature is concentrated near the wall.

them as M increases, it is readily seen that, as M increases, the surface temperature rises significantly when close to the wall, and then it levels out in the bulk pool. Figure 22 shows a plot of a steady-state surface temperature value versus distance from the wall for various values of M. A particular value of the temperature (in this case, T = 0.500) is used in the comparison. Using a semilog plot, as M increases, the distance from the wall at which the surface temperature is equal to 0.500 decreases. This graph can be used to model the surface temperature behavior for any Marangoni number in the range of 1 to 300. Given T = 0.500 and a Marangoni number, interpolation can be used to predict how close to the wall this temperature

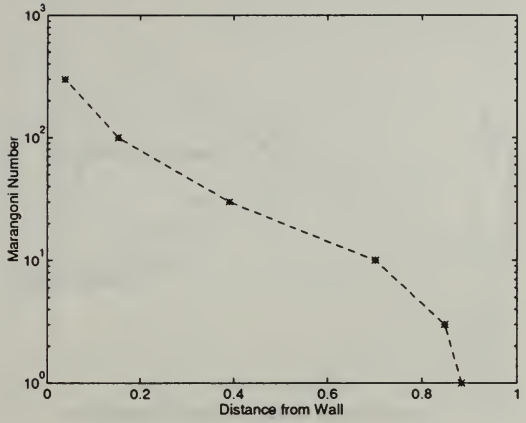

Figure 22. Distance from Wall with Steady-State Surface Temperature = 0.5: This plot takes a constant steady-state temperature and compares the Marangoni number's effect on distance from the wall. As M increases, the surface isotherm = 0.5, moves closer to the wall.

value will occur, providing information on the thermal field. The heat transfer at the wall is proportional to T_x, the derivative of the temperature solution with respect to the x-direction. Another useful modeling tool is the comparison of the temperature gradient at the wall as a function of Marangoni number. Figure 23 shows a log-log plot of the inverse surface temperature gradient out of the cold corner into the flow field versus Marangoni number. It is readily seen that as the value of M increases, the slope of the inverse temperature gradient at the corner also increases.

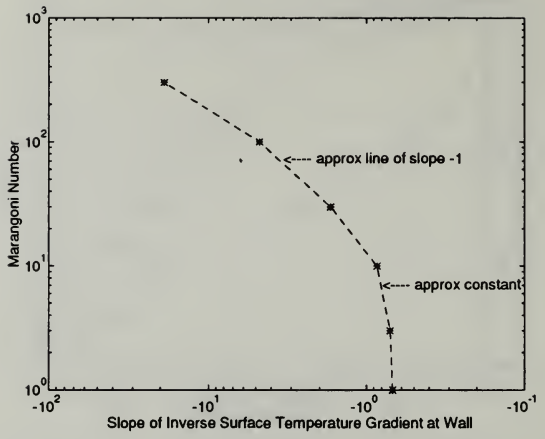

Figure 23. Steady-State Inverse Surface Temperature Gradient into Cold Corner: With low M, the slope is approximately constant; however, as M increases beyond 10, the slope of the line joining data points is approximately -1.

C. COMPARISON OF SURFACE VELOCITY VS MARANGONI NUMBER

Figure 24 shows a comparison of steady-state surface velocity profiles for various Marangoni numbers. All of the curves show a maximum value close to 0.2, except for that curve where $M = 300$. The reason for this is the afore-mentioned lack of resolution which arose by using a grid spacing of $h = 0.02$, which was not fine enough. From this figure, as the value of M increases, the position of this maximum velocity "moves" towards the wall. Also, the velocity appears to drop off after this maximum point. For higher Marangoni values, this decay is more rapid. For example, with $M = 10$, the maximum surface velocity value is 0.2058, which occurs at a distance

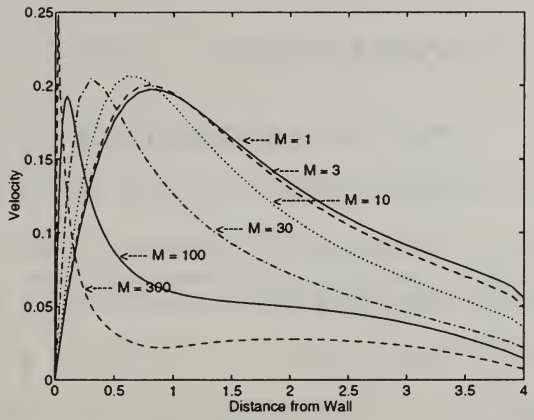

Figure 24. Steady-State Surface Velocity Profiles for Various M Values: As M increases, the surface velocity exhibits a sharper rise to its maximum value (closer to the wall), and a steeper descent into the bulk pool as well.

of 0.7 units from the wall (corner). With $M = 100$, the peak surface velocity value is 0.1927, but it occurs at a distance of only 0.1 units from the wall. Figure 25 shows a plot of peak surface velocity values versus distance from the wall for various Marangoni numbers. This plot can be used to model surface velocity behavior for any Marangoni number in the range of 1 to 300.

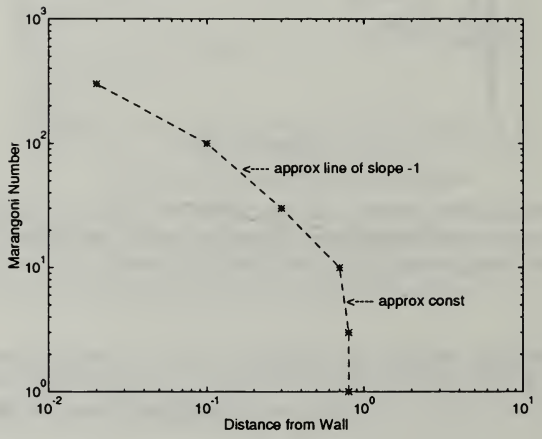

Figure 25. Distance from Wall with Peak Surface Velocity: This plot compares maximum velocity for various M values. As M increases, the distance from the wall at which the maximum is reached is smaller. Also, for small $M < 10$, the distance appears constant. For $M > 10$, the slope of the line connecting data points is approximately -1.

VIII. CONCLUSIONS

A. USE OF THE GREEN'S FUNCTION

Recall that, when inertia forces are negligible, a stream function is conveniently introduced to satisfy the conservation of mass equation (II.1). The flow depends everywhere only on the instantaneous thermal gradient along the surface, and it has been shown that the flow field can be represented using the Green's function for a point force near a rigid wall (see III.9), directed toward the wall. Velocity components in two dimensions can then be calculated and the time-dependent equation for the temperature field (using the ADI scheme) can be integrated to reach steady-state conditions. This is possible since the scaling factors (horizontal and vertical lengths and velocity) are of order unity [Ref. 20]. This means that when $R \ll 1$, the scaling used in the nondimensionalization is appropriate throughout the domain, implying that the solution is nearly independent of these parameters. If inertia were not neglected, this could not be possible, as resistance caused by inertial forces could reduce the velocity and length scales, as noted in [Ref. 20]. Therefore, although the numerical integration techniques employed in calculating the velocity components require substantial computational resources, the Green's function method of studying and solving the problem is a valid one. For example, one advantage of the Green's function is that the flow can still be represented over the entire quarter plane, so that there is no artificial recirculation due to the imposed artificial boundaries. The Green's function

approach does not require flow boundary conditions at the computational domain's boundaries. The flow is assumed isothermal across the boundary and is recirculating, decaying with distance.

B. CONDUCTIVE VS CONVECTIVE REGIME

As seen in the steady-state temperature distributions for the various values of Marangoni number (ranging $M = 1$ to $M = 300$), it appears that the flow is affected by thermal convection in the cases where $M \geq 30$. In Figure 16 ($M = 10$), there is slight evidence of convective effects. With $M = 30$, the isotherms are beginning to form a compressed region near the cold corner, and the stronger evidence of convective effects is apparent. Away from the wall, the isotherms are beginning to fan out in the shape of an ellipse, directed away from the wall. Figures 23 and 25 show an apparently inverse relationship of Marangoni number with inverse surface temperature gradient and peak surface velocity position, respectively, as $M \geq 10$. Even in the highly convective cases (large M value), no thermal boundary layers formed, although the isotherms were compressed.

C. LIMITATIONS AND FUTURE WORK

As mentioned earlier, The *MATLAB* routines could not accurately evaluate cases where $M > 300$, due to resolution problems. The grid spacing, h, needs to be reduced significantly as M increases. One solution is to increase the amount of temporary storage space which the computer uses in running *MATLAB*. An alternate

method would be to rewrite the numerical analysis code in a more efficient programming language, such as C, or to use another implicit method. Stone's strongly implicit method [Ref. 21] calculates all of the next iterative values by a direct elimination method, assuming that the successive vectors of iterative values will approximate the exact solution vector very closely after only a few iterations. This method solves a banded tridiagonal matrix and is economical arithmetically in relation to older methods. Also, the rate of convergence is much less sensitive to the choice of iteration parameters than the ADI method [Ref. 21]. In addition, using a nonuniform grid spacing (where h is smaller near the cold corner) might enable a more accurate simulation. One more technique is to model the convective behavior with the Green's function as a function of distance from the cold wall. As the Marangoni number increases, move the artificial boundaries closer to the wall. This may affect the flow geometry and convective effects.

The importance of the temperature and velocity distributions obtained from this research is an indicator that further work is needed. Figure 24 correlates well with the work of Zehr, et al, as explained by Chen [Ref. 1], comparing the shapes of the expanded surface velocity distribution curves at the cold corner. This research simplified the actual problem, revising the geometry and ignoring material properties. The free surface, which was assumed flat due to surface tension, might in fact not be flat. Also, the sudden change in temperature at the unit depth (from $T = -1$ to $T = 0$) is a good first approximation, but the shape of the molten metal/solid metal

interface might affect this change. Future research should concentrate on accurately predicting the effects of geometry change.

D. MODELING MATERIALS PROCESSING

This research emphasizes the importance of further studying the cold corner problem. In the broad arena of materials processing, an increase in the understanding of the forces that drive convective behavior (thermocapillary, bouyant, or electromagnetic) is essential. Since convection in the molten metal pool can significantly affect the microstructure of the finished material, theoretical and analytical work in this field should be continued.

REFERENCES

[1] M. M. Chen. Thermocapillary convection in materials processing. In S. K. Samanta et al., editors, *Interdisciplinary Issues in Materials Processing and Manufacturing (ASME)*, pages 541–557, 1987.

[2] S. Ostrach. Low-gravity fluid flows. *Ann. Rev. Fluid Mech.*, 14:313–345, 1982.

[3] J. F. Lancaster, editor. *The Physics of Welding.* International Institute of Welding, Pergamon Press, second edition, 1986.

[4] C. L. Chan, J. Mazumder, and M. M. Chen. Effect of surface tension gradient driven convection in a laser melt pool: Three-dimensional perturbation model. *J. Appl. Phys.*, 64 (11):6166–6174, December 1988.

[5] W. F. Hughes and J. A. Brighton. *Fluid Dynamics.* Schaum's Outline Series of Theory and Problems. McGraw-Hill, Inc., New York, second edition, 1991.

[6] A. Zebib, G. M. Homsy, and E. Meiburg. High marangoni number convection in a square cavity. *Phys. Fluids*, 28 (12):3467–3476, December 1985.

[7] L. Prandtl. *Essentials of Fluid Dynamics (Authorized Translation).* Hafner Publishing Company, New York, 1952.

[8] O. Reynolds. *Proc. Lit. and Phil. Soc. Manchester*, 14, 1874-75.

[9] O. Reynolds. Papers on mechanical and physical subjects. 1:81, 1900.

[10] L. Prandtl. *Phys. Zeitschr.*, 11:1072, 1910.

[11] S. J. Cowley and S. H. Davis. Viscous thermocapillary convection at high marangoni number. *J. Fluid Mech.*, 135:175–188, 1983.

[12] G. K. Batchelor. *An Introduction to Fluid Dynamics.* Cambridge University Press, Cambridge, 1967.

[13] F. B. Hildebrand. *Methods of Applied Mathematics.* Prentice-Hall, Inc., New Jersey, 1965.

[14] J. R. Blake. A note on the image system for a stokeslet in a no-slip boundary. *Proc. Camb. Phil. Soc.*, 70:303–310, 1971.

[15] H. Hasimoto and O. Sano. Stokeslets and eddies in creeping flow. *Ann. Rev. Fluid Mech.*, 12:35–363, 1980.

[16] R. V. Churchill and J. W. Brown. *Complex Variables and Applications.* McGraw-Hill, Inc., New York, 1990.

[17] IMSL Problem-Solving Software Systems. *User's Manual, MATH/LIBRARY* FORTRAN *Subroutines for Mathematical Analysis*, 1987. pp. 688–693.

[18] R. F. Boisvert. A fourth order accurate fast direct method for the helmholtz equation. In G. Birkoff and A. Schoenstadt, editors, *Elliptic Problem Solvers II*, pages 35–44. Academic Press, Inc., New York, 1984.

[19] D. A. Anderson, J. C. Tannehill, and R. H. Pletcher. *Computational Fluid Mechanics and Heat Transfer.* Hemisphere Publishing Corporation, New York, 1984.

[20] D. Canright. Thermocapillary flow near a cold wall. Technical Report NPS-MA-93-011, Naval Postgraduate School, January 1993.

[21] G. D. Smith. *Numerical solution of partial differential equations.* Oxford Applied Mathematics and Computing Science Series. Clarendon Press, Oxford, third edition, 1985.

APPENDIX A. FORTRAN ROUTINE

```
*   PROGRAM PDESOL

*   THIS IS A FORTRAN PROGRAM TO SOLVE POISSON'S EQUATION:
*   T-XX + T-YY = M (U-X * T-X + U-Y * T-Y)

    INTEGER   NCVAL, NX, NXTABL, NY, NYTABL
    PARAMETER (NCVAL=11, NX=5, NXTABLE=5, NY=4, NYTABL=4)

    INTEGER   I, IBCTY(4), IORDER, J, NOUT
    REAL      AX, AY, BRHS, BX, BY, COEFU, ERROR, FLOAT, PRHS, QD2VL,
   &          TRUE, U(NX,NY), UTABL, X, XDATA(NX), Y, YDATA(NY),
   &          MATHX, MATHY, M
    INTRINSIC FLOAT
    EXTERNAL  BRHS, FPS2H, PRHS, QD2VL, UMACH
    COMMON RHS(0:4, 0:3)
    PRINT *, 'ENTER MARANGONI NUMBER'
    READ *, M
    OPEN(UNIT = 10, FILE = 'DAVE1 DATA', STATUS = 'OLD')
    OPEN(UNIT = 15, FILE = 'OUTPUT DATA', STATUS = 'OLD')
    DO 5 I = 0, 4
    DO 5 J = 0, 3
      READ(10, *) MATHX, MATHY, RHS(I, J)
      RHS(I, J) = M * RHS(I, J)
      WRITE(15, 6) RHS(I, J)
  5 CONTINUE
  6 FORMAT(5X, F12.7)
*
*         SET RECTANGLE SIZE
*
    AX = 0.0
    BX = 4.0
    AY = 0.0
    BY = 3.0
*
*         SET BOUNDARY CONDITION TYPES
*
```

```
      IBCTY(1) = 1
      IBCTY(2) = 2
      IBCTY(3) = 1
      IBCTY(4) = 1
*
*           COEFFICIENT OF U (FOR HELMHOLTZ EQUATION)
*
      COEFU = 0.0
*
*           ORDER OF THE METHOD (2 OR 4)
*
      IORDER = 4
*
*           SOLVE THE PDE
*
      CALL FPS2H (PRHS, BRHS, COEFU, NX, NY, AX, BX, AY, BY, IBCTY,
     &     IORDER, U, NX)
*
*           SETUP FOR QUADRATIC INTERPOLATION
*
      DO 10 I = 1, NX
        XDATA(I) = AX + (BX - AX) * FLOAT(I-1) / FLOAT(NX-1)
   10 CONTINUE
      DO 20 J = 1, NY
        YDATA(I) = AY + (BY - AY) * FLOAT(J-1) / FLOAT(NY-1)
   20 CONTINUE
*
*           PRINT THE SOLUTION
*
      CALL UMACH (2, NOUT)
      WRITE (NOUT, '(8X,A,11X,A,11X,A,8X,A)') 'X', 'Y', 'U', 'ERROR'
      OPEN(UNIT = 20, FILE = 'GR1OUT DATA', STATUS = 'OLD')
      DO 40 J = 1, NYTABL
        DO 30 I = 1, NXTABL
          X     = AX + (BX - AX) * FLOAT(I-1) / FLOAT(NXTABL-1)
          Y     = AY + (BY - AY) * FLOAT(J-1) / FLOAT(NYTABL-1)
          UTABL = QD2VL (X,Y,NX,XDATA,NY,YDATA,U,NX,.FALSE.)
          TRUE  = 0.0
          ERROR = TRUE - UTABL
          WRITE (NOUT, '(4F12.7)') X, Y, UTABL, ERROR
          WRITE (20, '(4F12.7)') X, Y, UTABL, ERROR
   30   CONTINUE
```

```
 40 CONTINUE
    END
*
    REAL FUNCTION PRHS (X, Y)
    REAL    X, Y
    INTEGER IX, IY
    COMMON RHS(0:4, 0:3)
*
*         DEFINE RIGHT-HAND SIDE OF THE PDE
*
    IX = NINT(X)
    IY = NINT(Y)
    PRHS = RHS(IX, IY)
    RETURN
    END
*
    REAL FUNCTION BRHS (ISIDE, X, Y)
    INTEGER ISIDE
*
*         DEFINE THE BOUNDARY CONDITIONS
*
    IF (ISIDE .EQ. 2) THEN
      BRHS = 0.0
    ELSEIF (ISIDE .EQ. 3) THEN
      IF (0 .GT. Y .AND. Y .LT. 1) THEN
        BRHS = -1.0
      ELSE
        BRHS = 0.0
      ENDIF
    ELSE
      BRHS = 0.0
    ENDIF
    RETURN
    END
```

APPENDIX B. MATLAB ROUTINES

1. ADI.M

```
function [tnew, steps, u, v] = adi(hx, hy, M, num)
% PROGRAM adi.m
%
% This program uses the ADI method to solve the unsteady 2D convection -
% diffusion equation:
%
%                                          2        2
%       d T        d T        d T        d  T     d  T
%   M ( --- + u  --- + v  --- ) =  ----  +  ----
%       d t        d x        d y          2        2
%                                        d x      d y
%
% with the boundary conditions:
%     T(0, y) =  1 when 0 <= y <= 1
%     T(0, y) =  0 when 1 <= y
%     T(L, t) =  0 (L -> oo)
%    Ty(x, 0) =  0 (at the surface)
% as well as the initial guess for T(x, y):
%     T(x, y) =  (1 / Pi) * ArcTan [(2 x) / (x^2 + y^2 -1)]
%
% The program uses two subprograms for computing the LU decomposition of a
% tridiagonal matrix and then solving L U x = b.  A system of equations then
% result of the form (a two-step process):
%
%                      *          n
% (STEP 1):      A * T    = B * T
%                      x,y        x,y
%
%
%                      n+1        *
% (STEP 2):      C * T    = D * T
%                      x,y        x,y
%
% where A, B, C, and D are all tridiagonal matrices.
%
```

```
% The input arguments are:  hx     the spatial step in the x-direction
%                           hy     the spatial step in the y-direction
%                           M      the Marangoni number (a constant)
%
% The output is T, the temperature solution to the unsteady 2D convection -
% diffusion equation.
%
% The PDE will be solved using a two-step alternating-direction implicit
% (ADI) method, taken from Computational Fluid Mechanics and Heat Transfer,
% by Anderson, Tannehill, and Pletcher, p.167.
%
% NOTATION:  An indexed pair (i,j) corresponds to (row, column), which
%            further corresponds to (y-direction, x-direction).
%            Index(1,1) corresponds to the surface row, wall column.
%
% To satisfy the scheme's stability requirement, use a Courant number of .4

dt = .4 * hx * M;
steps = 0;
Nx = 4 / hx;
Ny = 3 / hy;
temp = tinit(hx, hy);
temp(:,Nx+1) = zeros(Ny+1,1);
temp(Ny+1,:) = zeros(1,Nx+1);
temp((1/hy)+1,1) = .5;
temp
k = [0 : hx : 4];

for t = 1:num;
  d25 = temp(1,:);
  tderiv = ones(1,Nx+1);
  for i = 1:Nx+1;
    if i == 1,
       tderiv(1) = (d25(2) - d25(1)) / hx;
    elseif i == Nx+1,
       tderiv(Nx+1) = (d25(Nx+1) - d25(Nx)) / hx;
    else
       tderiv(i) = (d25(i+1) - d25(i-1)) / (2 * hx);
    end;
  end;
  u = zeros(Ny+1,Nx+1);
  v = zeros(Ny+1,Nx+1);
```

```
  for j = 1:Nx+1;
    for i = 1:Ny+1;
        x = (j - 1) * hx;
        y = (i - 1) * hy;
        u(i,j) = ( trap((tderiv(j) .* uderiv(x, y, k, hx)), hx) );
        v(i,j) = ( trap((tderiv(j) .* vderiv(x, y, k)), hx) );
    end
  end
  v(1,:) = zeros(1,Nx+1);
  v(:,1) = zeros(Ny+1,1);
  u(:,1) = zeros(Ny+1,1);

  xcon1 = dt / (M * hx^2);
  xcon2 = dt / (4 * hx);
  ycon1 = dt / (M * hy^2);
  ycon2 = dt / (4 * hy);

% A and B are given by:

  c1 = (xcon2 .* u) - (xcon1 / 2);
  c2 = 1 + xcon1;
  c3 = (-xcon2 .* u) - (xcon1 / 2);
  c4 = (-ycon2 .* v) + (ycon1 / 2);
  c5 = 1 - ycon1;
  c6 = (ycon2 .* v) + (ycon1 / 2);

% where
%
%     *           *          *           n           n           n
% c1 T      + c2 T    + c3 T      = c4 T      + c5 T     + c6 T
%     x+1,y       x,y       x-1,y       x,y+1       x,y         x,y-1
%
% To set the boundary condition at the wall:

  c1(:,1) = zeros(Ny+1,1);
  c4(:,1) = zeros(Ny+1,1);
  c6(:,1) = zeros(Ny+1,1);

% To set the boundary conditions at the far right and bottom:

  c1(Ny+1,:) = zeros(1,Nx+1);
  c3(Ny+1,:) = zeros(1,Nx+1);
```

```
  c3(:,Nx+1) = zeros(Ny+1,1);
  c6(Ny+1,:) = zeros(1,Nx+1);

% To set the surface Neumann condition:

  c4(1,2:Nx) = ycon1 * ones(1,Nx-1);

  tempa = zeros(Ny+1,Nx+1);
  tempa = [(c4(1:Ny,:) .* temp(2:Ny+1,:)); zeros(1,Nx+1)] + (c5 .* temp) ...
        + [zeros(1,Nx+1); (c6(2:Ny+1,:) .* temp(1:Ny,:))];
  tempn = reshape(tempa', (Ny+1)*(Nx+1), 1);

  c1a = [c1(:,1:Nx) zeros(Ny+1,1)];
  c2a = c2 * ones(Ny+1,Nx+1);
  c2a(:,1) = c5 * ones(Ny+1,1);
  c3a = [zeros(Ny+1,1) c3(:,2:Nx+1)];
  c1new = reshape(c1a', (Ny+1)*(Nx+1), 1);
  c2new = reshape(c2a', (Ny+1)*(Nx+1), 1);
  c3new = reshape(c3a', (Ny+1)*(Nx+1), 1);

  A = [c3new c2new c1new];
  A(1,1) = NaN;
  A((Ny+1)*(Nx+1),3) = NaN;

% Use crout(A) to compute the LU factorization of a tridiagonal matrix A.

  [lower, upper] = crout(A);

% Use croutslv(lower, upper, Trhs) to solve L U T* = Tn, where A = L U is
% tridiagonal, Tn is the known temperature, and T* is the step temperature.

  tst = zeros(size(A(:,1)));
  tst = croutslv(lower, upper, tempn);
  tsta = reshape(tst, Nx+1, Ny+1);
  tstar = tsta';

% Now we repeat the operation in the other direction (STEP 2).
%
% C and D are given by:

  d1 = (ycon2 .* v) - (ycon1 / 2);
  d2 = 1 + ycon1;
```

```
  d3 = (-ycon2 .* v) - (ycon1 / 2);
  d4 = (-xcon2 .* u) + (xcon1 / 2);
  d5 = 1 - xcon1;
  d6 = (xcon2 .* u) + (xcon1 / 2);

% where
%
%     n+1          n+1        n+1          *           *           *
% d1 T      + d2 T     + d3 T       = d4 T      + d5 T      + d6 T
%     x,y+1        x,y        x,y-1        x+1,y       x,y         x-1,y
%
% To set the boundary condition at the wall:

  d1(1:Ny,1) = zeros(Ny,1);
  d3(2:Ny+1,1) = zeros(Ny,1);
  d4(:,1) = zeros(Ny+1,1);

% To set the boundary conditions at the far right and bottom:

  d1(1:Ny,Nx+1) = zeros(Ny,1);
  d3(Ny+1,:) = zeros(1,Nx+1);
  d3(2:Ny+1,Nx+1) = zeros(Ny,1);
  d6(1:Ny,Nx+1) = zeros(Ny,1);

% To set the surface Neumann condition:

  d1(1,2:Nx+1) = (- ycon1) .* ones(1,Nx);

  tstarn = zeros(Ny+1,Nx+1);
  tstarn = [(d4(:,1:Nx) .* tstar(:,2:Nx+1)) zeros(Ny+1,1)] + (d5 .* tstar)..
         + [zeros(Ny+1,1) (d6(:,2:Nx+1) .* tstar(:,1:Nx))];
  tstaro = reshape(tstarn, (Ny+1)*(Nx+1), 1);

  d1a = [d1(1:Ny,:); zeros(1,Nx+1)];
  d2a = d2*ones(Ny+1,Nx+1);
  d2a(:,1) = d5 * ones(Ny+1,1);
  d3a = [zeros(1,Nx+1); d3(2:Ny+1,:)];
  d1new = reshape(d1a, (Ny+1)*(Nx+1), 1);
  d2new = reshape(d2a, (Ny+1)*(Nx+1), 1);
  d3new = reshape(d3a, (Ny+1)*(Nx+1), 1);

  C = [d3new d2new d1new];
```

```
  C(1,1) = NaN;
  C((Ny+1)*(Nx+1),3) = NaN;

% Use crout(C) to compute the LU factorization of a tridiagonal matrix C.

  [unter, oben] = crout(C);

% Use croutslv(unter, oben, tstarp) to solve L U Tnew = T*, where C = L U is
% tridiagonal, T* is the known temperature, and Tnew is the step temperature.

  tplus = zeros(size(C(:,1)));
  tplus = croutslv(unter, oben, tstaro);
  tnew = reshape(tplus, Ny+1, Nx+1);
  n = norm(reshape((tnew - temp), (Ny+1)*(Nx+1), 1));
  if n < .0001, break, return;
  end;
  steps = steps + 1;
  temp = tnew;

  steps
end
return
```

2. TINIT.M

```
function out = tinit(hx, hy)
Nx = 4 / hx;
Ny = 3 / hy;
for j = 1:Nx+1;
    for i = 1:Ny+1;
        x = (j - 1) * hx;
        y = (i - 1) * hy;
        out(i,j) = (1/pi) * atan2((2*x),(x^2 + y^2 - 1));
    end;
end;
return;
```

3. TRAP.M

```
function out = trap(f, hk)
[M, N] = size(f);
out = (hk / 2) * ( f(1) + (2 * sum(f(2:N-1))) + f(N) );
return;
```

4. UDERIV.M

```
function out = uderiv(x, y, k, h)
out = (((4*x*y^2) .* k) ./ (((x + k).^2) + y^2).^2 -   ...
  (2*x .* k) ./ ((x + k).^2 + y^2) - (4*x*y^2 .* k) ./   ...
  (k.^4 - ((2*x^2) .* k.^2) + x^4 + ((2*y^2) .* k.^2) ...
  + 2*(x*y)^2 + y^4) + ...
  log(((x + k).^2 + y^2).^(1/2) ./ ((x - k).^2 + y^2).^(1/2))) ./ (2 * pi);
if y == 0,
  out(find( isnan(out) | out > 1)) = ...
  (.5 * log((2*x)+h) + log((2*x)-h) - log(h^2) + ...
  (2 * x^2) * (12 * x^2 - 5 * h^2) / ((4 * x^2) - h^2)^2) / (2 * pi);
end;
return;
```

5. VDERIV.M

```
function out = vderiv(x, y, k)
out = (-4*x*y .* k.^2 .* (k.^2 - x^2 + y^2)) ./ ...
  (3.1415 .* (k.^2 - 2*x .* k + x^2 + y^2) .* (k.^2 + 2*x .* k ...
  + x^2 + y^2).^2);
return;
```

6. CROUT.M

```
function [l, u] = crout(A)
%
% function [l, u] = crout(A)
%
% This function computes the LU factorization of a tridiagonal matrix A.
% A is input as an Nx3 matrix.  For example, if N = 5, the storage is:
%
%  _           _                  _              _
% ! NaN b1 c1  !                 ! b1 c1          !
% !  a2 b2 c2  !                 ! a2 b2 c2       !
% !  a3 b3 c3  !  represents     !    a3 b3 c3    !
% !  a4 b4 c4  !                 !       a4 b4 c4 !
% !_ a5 b5 NaN_!                 !_         a5 b5_!
%
% The variables returned are the Nx2  matrices l and u:
%
%  _      _    _    _          _                _    _                 _
% !  0 e1 !   ! 1 g1 !        ! e1               !   ! 1 g1             !
% ! d2 e2 !   ! 1 g2 !        ! d2 e2            !   !   1  g2          !
% ! d3 e3 !   ! 1 g3 !  for   !    d3 e3         !   !      1  g3       !
% ! d4 e4 !   ! 1 g4 !        !       d4 e4      !   !         1  g4    !
% !_d5 e5_!   !_1  0_!        !_         d5 e5_! !_             1 _!
%
%
[N, M] = size(A);
l = zeros(N,2);
u = l;
u(:,1) = ones(N,1);
l(2:N,1) = A(2:N,1);
l(1,2) = A(1,2);
u(1,2) = A(1,3)/l(1,2);
for j = 2:N-1,
    l(j,2) = A(j,2) - A(j,1) * u(j-1,2);
    u(j,2) = A(j,3) / l(j,2);
end
l(N,2) = A(N,2) - A(N,1) * u(N-1,2);
return
```

7. CROUTSLV.M

```
function [x, y] = croutslv(l, u, b)
%
% function [x, y] = croutslv(l, u, b)
%
% This program solves L U x = b, where A = L U is tridiagonal, by solving
% L y = b, and then solving U x = y.
% l and u are provided by crout.m
%
%
[N, m] = size(l);
y = zeros(N,1);
x = y;
y(1) = b(1) / l(1,2);

for j = 2:N,
    y(j) = (b(j) - l(j,1) * y(j-1)) / l(j,2);
end

x(N) = y(N);
for j = N-1:-1:1,
    x(j) = y(j) - u(j,2) * x(j+1);
end
return
```

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2. Library, Code 52 Naval Postgraduate School Monterey, CA 93943-5002	2
3. Professor David Canright Department of Mathematics Naval Postgraduate School Monterey, CA 93943-5000	2
4. Professor Clyde Scandrett Department of Mathematics Naval Postgraduate School Monterey, CA 93943-5000	1
5. Professor Van E. Henson Department of Mathematics Naval Postgraduate School Monterey, CA 93943-5000	1
6. Professor Richard Franke Department of Mathematics Naval Postgraduate School Monterey, CA 93943-5000	1
7. CPT Michael R. Huber Department of Mathematical Sciences United States Military Academy West Point, New York 10996-5000	3
8. Dr. George Yoder Office of Naval Research Materials Division 800 North Quincy Street Arlington, VA 22217-5660	3

9. Dr. Richard Lau 1
Office of Naval Research
Mathematical Sciences Division
800 North Quincy Street
Arlington, VA 22217-5660